TEKNILLINEN KORKEAKOULU
Automaatio- ja systeemitekniikan osasto

Tuomas Kaski

Taitoa vaativien tehtävien oppiminen ja suoritus WorkPartner -robotilla

Diplomityö, joka on jätetty opinnäytteenä tarkastettavaksi diplomi-insinöörin tutkintoa varten Espoossa 16.2.2004.

Työn valvojana ja ohjaajana toimi Professori Aarne Halme

TEKNILLINEN KORKEAKOULU
Automaatio- ja systeemitekniikan osasto

DIPLOMITYÖN TIIVISTELMÄ

Tekijä	Päiväys
Tuomas Kaski	10.2.2004
	Sivumäärä: 65

Työn nimi

Taitoa vaativien tehtävien oppiminen ja suoritus WorkPartner-robotilla

Professuuri	Koodi
Automaatiotekniikka	AS-84

Työn valvoja

Professori Aarne Halme

Työn ohjaaja

Professori Aarne Halme

Diplomityössä tutkitaan taitoa vaativien tehtävien opettamista ja suorittamista kehittyneelle palvelurobotille nimeltä WorkPartner. Työssä käsiteltävä taitoa vaativa tehtävä on "laatikon etsintä ja nosto" –tehtävä eli siinä robotti, opittuaan tehtävän, etsii tietyn laatikon ympäristöstään, menee sen luokse, nostaa laatikon ja vie sen määrättyyn kohteeseen.

Diplomityössä käytetään opettamismenetelmää, jossa robotin tietyissä osatehtävissä käytetään geneettistä algoritmia etsittäessä parempia ratkaisuja osatehtävien suoritukseen. Tämän algoritmin avulla robotin tehtävän suoritus paranee jokaisen tehtävän suoritusyrityksen jälkeen eli robotti kykenee oppimaan kokeilun ja erehdyksen avulla. Geneettinen algoritmi parantaa osatehtävien suoritusta muuttelemalla osatehtävissä olevien mikrotehtävien parametreja. Tämän lisäksi algoritmi pystyy lisäämään ja poistamaan mikrotehtäviä ja muuttamaan osatehtävissä olevien mikrotehtävien järjestystä, jonka ansiosta tehtävän suorituksen taso nousee huomattavasti.

Kun robotti saadaan oppimaan tehtävän suoritus, helpottuu vaikeiden ja monimutkaisten taitoa vaativien tehtävien tuteuttaminen. Tehtävän suunnittelijan ei tarvitse kuluttaa aikaa miettiäkseen, miten taitoa vaativa tehtävä pitää tarkalleen suorittaa. Riittää, että WorkPartnerille kerrotaan, mistä toiminnallisuuksista eli mikrotehtävistä taitoa vaativa tehtävä koostuu, annetaan robotille alkuarvot (satunnaiset tai mietityt) ja arvostellaan robotin suoritusyritykset fitness-arvoilla. Robotti tekee loput itse.

Tutkimuksen tuloksena saatiin ohjelma, jolla WorkPartnerille voidaan opettaa "laatikon etsintä ja nosto" –tehtävä. Ohjelmaa jatkokehittämällä voidaan toteuttaa ohjelmointialusta taitoa vaativille tehtäville. Tällöin yhdellä ohjelmalla pystyttäisiin hallitsemaan WorkPartnerin oppiminen.

Avainsanat	Kieli
geneettinen algoritmi, oppiminen, palvelurobotti, taitoa vaativa tehtävä	suomi

HELSINKI UNIVERSITY OF TECHNOLOGY
Department of Automation and System Engineering

ABSTRACT OF THE MASTER'S THESIS

Author	Date
Tuomas Kaski	10.2.2004
	Pages: 65

Title of Thesis
Learning and performing skill demanding tasks with WorkPartner robot

Chair	Chair Code
Automation Engineering	AS-84

Supervisor
Professor Aarne Halme

Instructor
Professor Aarne Halme

This thesis researches the learning and performing of skill demanding tasks with an advanced service robot called WorkPartner. The skill demanding task studied in this work composes of finding a box from the surroundings, moving to the box, lifting the box and taking the box to a specific place.

In this thesis a learning method was used, where a genetic algorithm is used in certain subtasks of the robot to seek better solutions for the subtasks. With this algorithm the robot's performance of a task gets better with each attempt at performing the task. This means that the robot can learn through trial and error. The genetic algorithm improves the performance of subtasks by changing the parameters of the microtask which are parts of the subtask. In addition to this the algorithm can add and remove microtasks and change the order in which the microtasks are executed within the subtask. This increases the performance of the skill demanding task considerably.

When a robot can learn to perform a task, making difficult and complicated skill demanding tasks gets easier. The designer of a task doesn't have to waist time trying to come up with the perfect solution to the task. It is enough if WorkPartner is told what components (microtasks) the task has, it is given starting values (random or specific) and it's task performance is ranked with a fitness value. The robot does the rest.

This research produced a computer program that is used to teach the "find and lift box" –task to WorkPartner. By further developing the program, a programming platform can be made for producing other skill demanding tasks. This one platform could control all learned tasks of WorkPartner.

Keywords	Language
genetic algorithm, learning, service robot, skill demanding task	Finnish

Alkulause

Tämä diplomityö on tehty Teknillisen korkeakoulun Automaatiotekniikan laboratoriossa osana palvelurobottia kehittävää projektia.
Kiitän työni valvojaa professori Aarne Halmetta mielenkiintoisesta ja monipuolisesta diplomityöpaikasta. Esitän kiitokseni diplomi-insinööri Mikko Heikkilälle, diplomi-insinööri Jouni Sievilälle ja filosofian maisteri Maija Kaskelle arvokkaista neuvoista ja hyvästä yhteistyöstä. Lisäksi kiitän kaikkia niitä henkilöitä, jotka ovat auttaneet minua työni valmistumisessa.

Espoossa helmikuussa 2004

Tuomas Kaski

Sisällys

Käytetyt merkinnät ja lyhenteet

CAN-väylä	Controller Area Network
GA	Genetic Algorithm (Geneettinen Algoritmi)
GPS	Global Positioning System
PTU	Pan-Tilt-Unit
QNX	Reaaliaika käyttöjärjestelmän nimi
RS-232	Sarja liitännän standardi
WLAN	Wireless Local Area Network
WoPa	WorkPartner-robotti
Yo-yo-ohjain	Teleoperointivaljaat

1 Johdanto

1.1 Tutkimuksen tausta

Robotit ovat suuresta kehityksestään huolimatta vielä hyvin alkeellisia laitteita, jotka osaavat tehdä ainoastaan niihin tarkkaan ohjelmoituja asioita. Roboteilta puuttuu luovuus, koska robotti ei kykene tekemään tekemään asioita, joita se ei ole ennen tehnyt. Luovuus mielletään usein kyvyksi luoda uutta (Robbins, 2001, s. 133). Mikä siis antaa ihmiselle hänen luovuutensa? Ainakin tämän kyky oppia. Olisiko mahdollista tehdä myös robotista oppivaa, edes pienessä määrin, jolloin robotin ongelmanratkaisukyky paranisi huomattavasti.

Palvelurobotiikan alalla näihin ongelmiin pyritään kehittämään ratkaisuja. Palvelurobottien olisi tarkoitus auttaa ihmistä hänen arkipäivän askareissaan ja ihmiselle tyypillisessä ympäristössä. Palvelurobotti rakennetaan usein tiettyä tehtävää varten, mutta WorkPartner-projektissa ongelmaa on lähdetty ratkaisemaan toiselta kannalta; rakentamalla robotti siten, että se pystyy suorittamaan monia erilaisia tehtäviä. Projektin ensimmäisessä vaiheessa robotille rakennettiin runko, jolla on mahdollista liikkua vaikeissakin oloissa. Sitten robotille rakennettiin ihmisen yläruumista muistuttava manipulaattoriyksikkö, jotta se pystyisi tekemään samanlaisia työtehtäviä kuin ihminen. Viimeisessä eli nyt käynnissä olevassa vaiheessa robotti pyritään ohjelmoimaan suorittamaan monimutkaisia tehtäviä ja lopulta jopa oppimaan työtehtäviä itse.

WorkPartner-projektissa tavoitteena on saada rakennettua mahdollisimman pitkälle automatisoitu kenttä- ja palvelurobotti, ja mikä olisikaan automaattisempi kuin itsestään oppiva robotti. Valmiille WorkPartnerille ei tarvitsisi yrittää selvittää yksityiskohtaisesti miten jokin tehtävä suoritetaan, vaan tehtävä voitaisiin vain antaa ja opettaa perusliikkeiltään, minkä jälkeen robotti pyrkisi itsenäisesti annettuun tavoitteeseen. Aluksi suorituksen taso olisi heikko, mutta onnistumisien myötä robotti saisi positiivista palautetta ympäristöstään tai ihmiseltä ja oppisi ennen pitkää suoriutumaan annetusta tehtävästä.

1.2 Diplomityö

Tässä diplomityössä esitellään opettamismenetelmä, jossa robotin tietyissä osatehtävissä käytetään geneettistä algoritmia etsittäessä parempia ratkaisuja

osatehtävien suoritukseen. Tämän algoritmin avulla robotin kokonaistehtävän suoritus paranee jokaisen tehtävän suoritusyrityksen jälkeen eli robotti kykenee oppimaan kokeilun ja erehdyksen avulla.

Tässä työssä on kehitetty menetelmä, jolla WorkPartner-robotti saadaan ilman valmiiksi ohjelmoituja paikkatietoja löytämään haluttu kohde, kulkemaan kohteen luo ja sen jälkeen käsittelemään kohdetta halutulla tavalla. Oletuksena on, että robotin mekaaninen liikkuvuus ja ympäristön esteettömyys mahdollistavat annetun tehtävän suorittamisen.

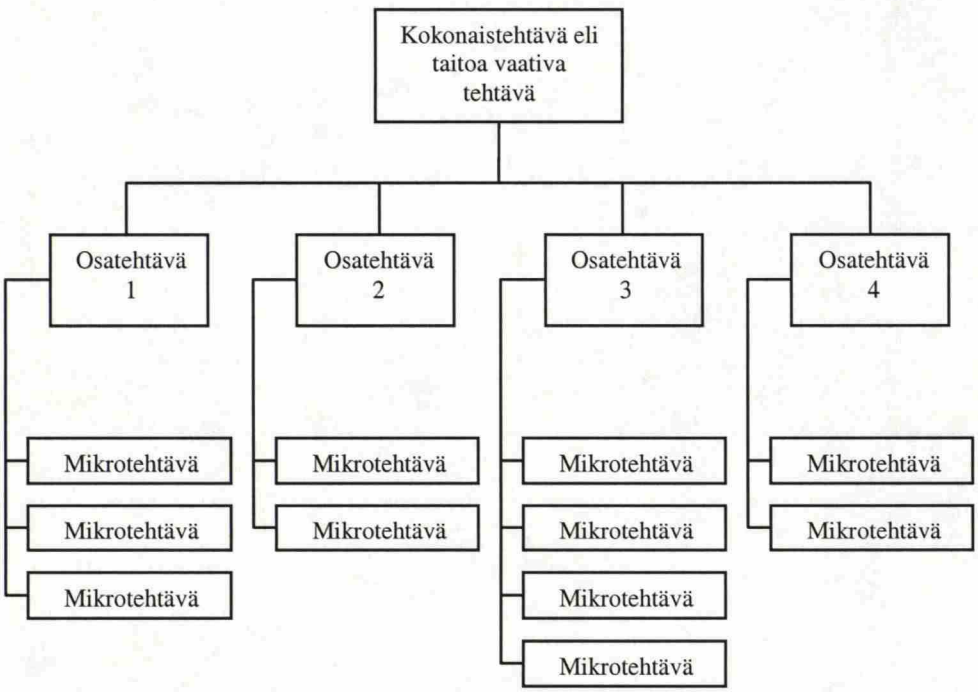
Esimerkkinä geneettisen algoritmin käytöstä oppimisessa esitellään taitoa vaativa tehtävä, jossa WorkPartner-robotti etsii ympäristöstään tehtävänannossa määriteltyä laatikkoa, kulkee sen luokse ja nostaa laatikon ylös. Kahdessa yllämainituista osatehtävistä, laatikon etsinnässä ja nostossa, käytetään oppimista hyväksi.

1.3 Taitoa vaativat tehtävät

Taitoa vaativat tehtävät eivät ole yksinkertaisia suorittaa, tarvitaan oppimista tai harjoittelua ennen kuin tehtävistä voi suoriutua. Tämä johtuu siitä, että ympäristö, jossa robotti toimii, on arvaamaton ja muuttuva. Vaihtuvaisen ympäristön tarkka mallintaminen on hyvin vaikeaa. Navigointi muuttuvassa ympäristössä on epätarkkaa, koska robotin on vaikea saada ympäristöstään luotettavaa tietoa. Sen siis pitäisi pystyä toimimaan niillä vähillä varmoilla tiedoilla, joita se ympäröivästä maailmastaan aistimillaan voi saada. Aistimilta saatava tieto vaikuttaa siihen, miten robotti tehtävänsä suorituttamista jatkaa. Tätä kutsutaan aistimilta saatavaksi takaisinkytkennäksi.

Taitoa vaativat tehtävät eli kokonaistehtävät koostuvat usein monesta pienemmästä osatehtävästä, jotka yksinään olisivat helpompia suorittaa. Osatehtävänä voisi olla esimerkiksi "etsi punainen laatikko kuvasta" tai "kulje paikkaan A". Kun osatehtäviä on monta, pitää yksittäisten osatehtävien suorituksen tukea muiden osatehtävien suorittamista, jotta kokonaistehtävästä selviytyminen onnistuisi. Esimerkiksi, jos kokonaistehtävä on "nouda punainen laatikko", osatehtävät voisivat tällöin olla "etsi punainen laatikko", "mene laatikon luokse", "nosta laatikko ilmaan" ja "tuo laatikko takaisin". Epäonnistuminen yhdessäkin vaiheessa eli osatehtävässä aiheuttaisi sen, että punaista laatikkoa ei pystyttäisi noutamaan.

Osatehtävät jakautuvat edelleen pienemmiksi suorituksiksi eli mikrotehtäviksi. Mikrotehtävät ovat pieniä toimia, kuten etäisyyden mittaus laserilla, robottialustan liike tai manipulaattorin siirto määriteltyyn paikkaa. Mikrotehtäviä voi osatehtävässä olla yksi tai useampia. Kokonaistehtävän jakaminen osiin ensin osatehtäviksi ja sitten mikrotehtäviksi mahdollistaa sen, että tehtävän rakenne on mahdollisimman selkeä, tehtävän opettaminen käy helposti ja osatehtävissä voidaan käyttää erilaisia oppimismenetelmiä tehtävän luonteesta riippuen. Alla olevassa kuvassa 1 on esitetty kokonaistehtävän jakautuminen tähän hierarkiseen muotoon.



Kuva 1. Taitoa vaativan tehtävän hierarkinen rakenne.

Taitoa vaativa tehtävä tarkoittaa sitä, että tehtävän suorittamiseen ei mikään aikaisemmin opituista toimintamalleista sellaisenaan riitä. Varsinkin, kun ympäristö jossa robotti toimii, on muuttuva ja arvaamaton, on robotin kyettävä havainnoimaan ympäristöään, kokeiltava tehtävään erilaisia ratkaisuja ja opittava niistä. Kulloinenkin ratkaisu on yksilöllinen eli se ei sellaisenaan käy muihin samantyyppisiin tilanteisiin. Robotin pitäisi kuitenkin kyetä hyödyntämään jo opitun tehtävän tietämystä suorittaessaan samaa kokonaistehtävää jossakin toisessa ympäristössä.

Taitavasti toimiminen tarkoittaa sitä, että robotilla on kyky suoriutua moniosaisista tehtäväkokonaisuuksista ilman että jokaisen osatehtävän parametrit on valmiiksi tarkasti ohjelmoitu. Robotilla ei esimerkiksi ole tiedossa noudettavan laatikon tarkkaa sijaintia, joten sen pitäisi osata löytää laatikko "etsimällä" ja hakea laatikko sieltä, missä se on. Kun robotti kykenee tähän, voidaan sanoa, että robotilla on taitoa suoriutua taitoa vaativista tehtävistä.

Tässä diplomityössä robotin taito löytää ja nostaa laatikko saadaan aikaan aluksi opettamalla ja sitten itseoppimalla. Robotille annetaan kokonaistehtävä, jota se yrittää suorittaa kokeilemalla erilaisia parametreja osatehtäviin ratkaisuiksi. Osatehtävien suoritus paranee sitä mukaa kun geneettinen algoritmi arvioi parametrien "hyvyyttä" tehtävän suorituksessa. Näin robotti oppii kokeilun ja erehdyksen (eriasteisten onnistumisten) kautta suorittamaan vaativia tehtäväkokonaisuuksia.

2 WorkPartner palvelurobotti

2.1 WorkPartnerin taustaa

Teknillisen korkeakoulun Automaatiotekniikan laboratoriossa on kehitetty vuodesta 1998 lähtien palvelurobottia nimeltä WorkPartner (kuva 2). Nimi WorkPartner tulee tavoitteesta saada aikaan sopeutuva ja oppiva palvelurobotti, joka pystyy käyttämään erilaisia työkaluja, työskentelemään ihmisten kanssa ja suorittamaan hyvin monenlaisia työtehtäviä. Robottia voidaan käskyttää läheltä puheella ja eleillä tai kauempaa teleoperoinnin ja internetin välityksellä. WorkPartner on suurehkon kokonsa (pituus 140 cm, leveys 120 cm, paino 240 kg) takia parhaiten hyödynnettävissä joko ulkona tai laajoissa sisätiloissa. Sille soveltuvia työtehtäviä ovat esimerkiksi kevyet metsä- ja puistotyöt, kuljetus- ja siivoustyöt sekä vartiointitehtävät (Leppänen et al. 1998).

Kuva 2. WorkPartner palvelurobotti.

2 WorkPartner palvelurobotti

2.1 WorkPartnerin taustaa

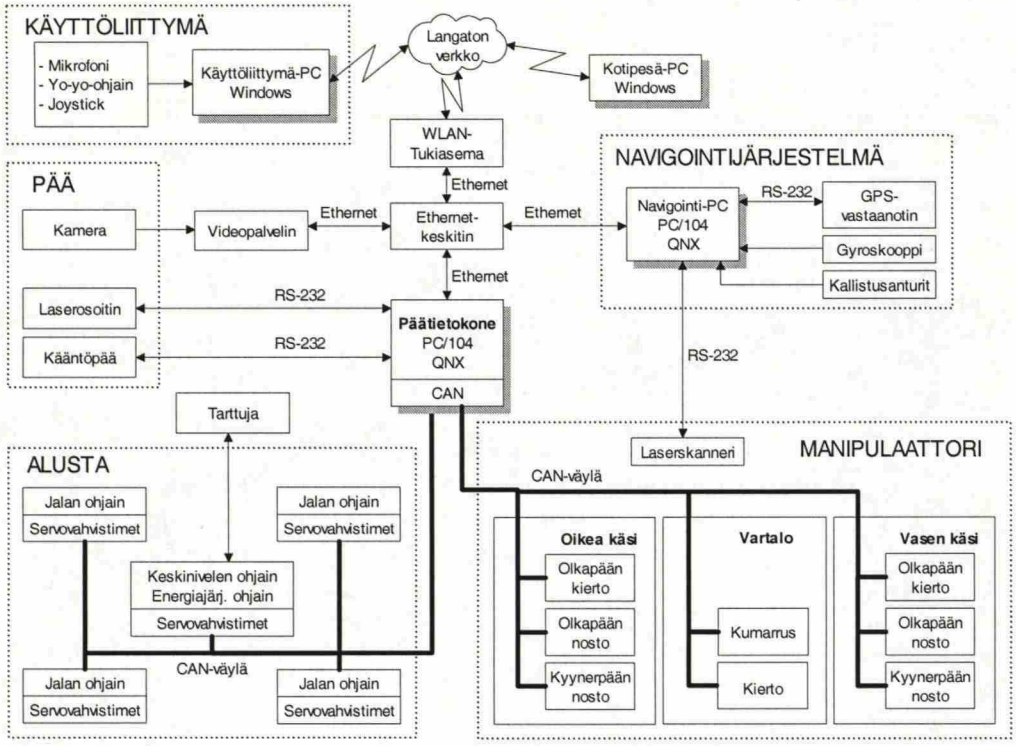
Teknillisen korkeakoulun Automaatiotekniikan laboratoriossa on kehitetty vuodesta 1998 lähtien palvelurobottia nimeltä WorkPartner (kuva 2). Nimi WorkPartner tulee tavoitteesta saada aikaan sopeutuva ja oppiva palvelurobotti, joka pystyy käyttämään erilaisia työkaluja, työskentelemään ihmisten kanssa ja suorittamaan hyvin monenlaisia työtehtäviä. Robotilta voidaan käskytää lähetä puheella ja eleillä tai kauempaa telekopterimin ja internetin välityksellä. WorkPartner on suurehkon kokoinen (pituus 140 cm, leveys 120 cm, paino 240 kg) tasia parhaiten hyödynnettävissä joko ulkona tai laajoissa sisätiloissa. Sille soveltuvia työtehtäviä ovat esimerkiksi kevyet mcrsä- ja puistotyöt, kuljetus- ja siivoustyöt sekä vartiointitehtävät (Eeppinen et al. 1998).

Kuva 2. WorkPartner palvelurobotti.

2.2 WorkPartnerin rakenne

WorkPartner koostuu runko-osasta, neljästä jalasta, manipulaattoriyksiköstä ja kahdesta manipulaattorista. Runko-osa on jaettu kahteen osaan ja osien välissä on keskinivel, joka mahdollistaa robotin kääntymisen sivusuunnassa. Rungon neljästä kulmasta lähtevät robotin jalat. Jokaisella jalalla on kolme vapausastetta ja lisäksi jalkojen päässä on pyörät. Jalat mahdollistavat robotin monipuolisen liikkumisen kävellen, pyörillä rullaten tai näiden yhdistelmällä pyöräkävelyllä (engl. "rolking"). WorkPartnerin rungon etuosassa on hieman ihmisen yläruumista muistuttava manipulaattoriyksikkö, johon on kiinnitetty kaksi ihmisen kättä muistuttavaa manipulaattoria (3 vapausastetta) ja kääntyvä pää (2 vapausastetta). Kääntyvässä päässä eli PTU:ssa (Pan Tilt Unit) on digitaalinen videokamera sekä 1-piste laseretäisyysmittalaite. Robotti muistuttaa kreikkalaisessa mytologiassa ollutta kentauri-olentoa, joka oli hevosen ja ihmisen yhdistelmä (Oksanen 2003).

WorkPartnerissa on useita laitteita, jotka välittävät mittaustietoa muille laitteille ja joille voidaan antaa komentoja. Kuvassa 3 on esitelty WorkPartnerin laitteet ja niiden välillä olevat tiedonsiirtoväylät. Tässä diplomityössä tehty ohjelma, joka suorittaa taitoa vaativia tehtäviä robotilla, sijaitsee käyttöliittymä-PC:llä. Käyttöliittymä-PC:n kautta ohjataan robotin muita laitteita.



Kuva 3. WorkPartnerin laitteistokaavio (Heikkilä M.).

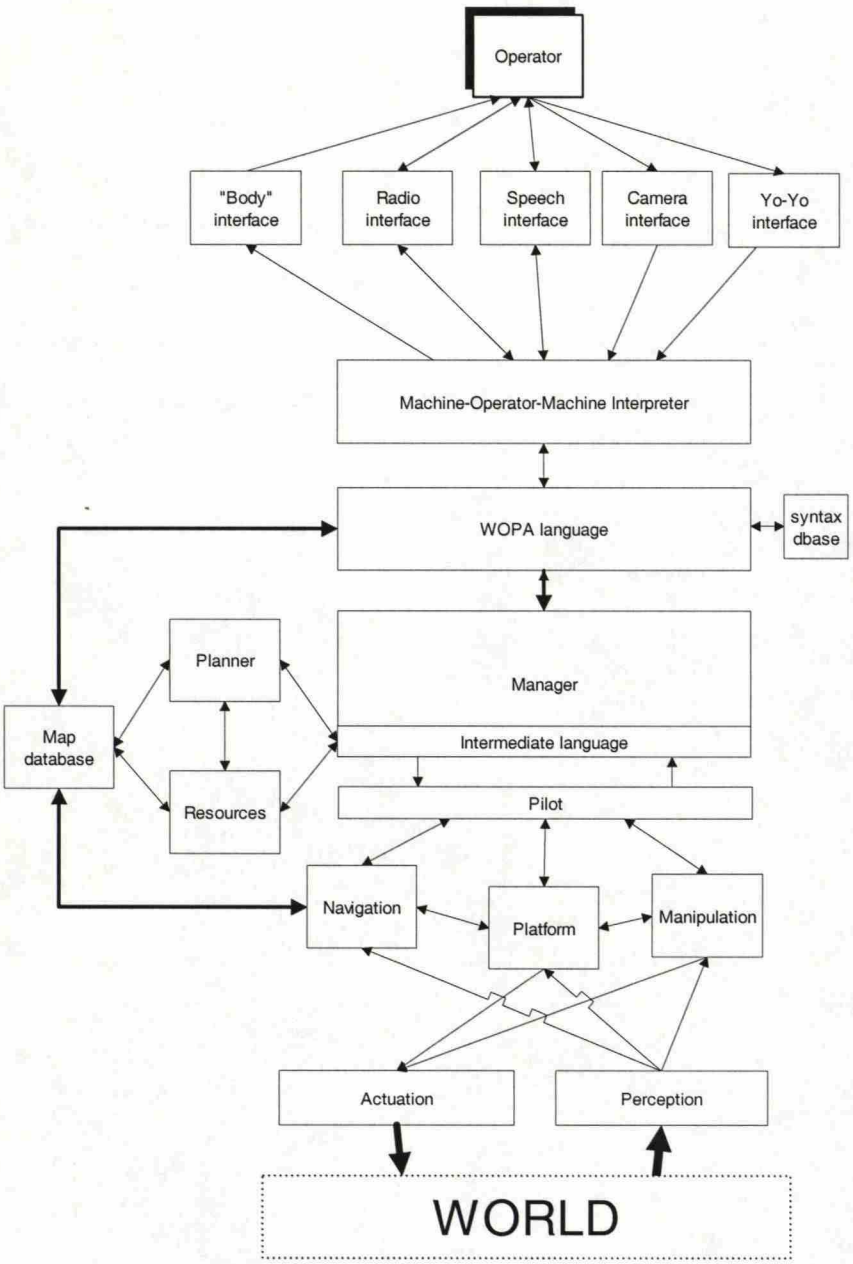
2.3 WorkPartnerin "äly"

Robotti on alunperin suunniteltu interaktiiviseksi, eli sellaiseksi, että se pystyy ymmärtämään sille annettuja käskyjä ja antaa palautetta omasta tilastaan. Tavoitteena on, että se käyttäytyy kuin koira, innokkaasti palvellen kunhan se ymmärtää mitä pitää tehdä. WorkPartnerin älykkyyttä onkin luonnollisempaa verrata koiraan kuin ihmiseen, juuri sen takia että robotin ja ihmisen välillä vallitsee samanlainen ymmärryksen vaikeus kuin koiran ja ihmisen välillä.

WorkPartnerin "äly" on hajautettu ja kerrostettu, niin että yksinkertaiset toiminnot, kuten jalan nivelen liikuttaminen, suoritetaan alimmalla tasolla ja monimutkaiset toiminnot, kuten "kulje paikkaan A", suoritetaan ylimmällä tasolla. Keskiniveleen ja jokaiseen jalkaan on liitetty mikrokontrolleri, joka hoitaa niveltoimilaitteiden säädöt. Mikrokontrollereita hallinnoi päätietokone, joka hoitaa ylemmän tason liikkeenohjauksen. Mikrokontrollerit ja päätietokone on yhdistetty CAN-väylällä keskenään. Manipulaattoriyksikköä ohjataan toisen CAN-väylän välityksellä pääkoneelta. Päätietokone, navigointitietokone, käyttöliittymä ja videopalvelin on

yhdistetty Ethernet-verkolla, joka jatkuu robotin ulkopuolelle WLAN-verkon läpi (Oksanen 2003).

WorkPartnerin koko ohjausarkkitehtuuri on esitetty kuvassa 4. Tämä arkkitehtuuri mahdollistaa käyttäjän ja robotin välisen vuorovaikutuksen. Käyttäjä antaa robotille tehtäviä korkeantason kielen välityksellä (kuvassa WoPa language). Kielen syntaksi sisältää komentoja ja objekteja, jotka liittyvät fyysiseen ympäristöön. Ohjelma nimeltä "manager" tulkitsee käyttäjän antamat viestit ja suunnittelee toimet tehtävän suorittamiseksi. Suunnittelun tuotos on (järjestetty) lista välikielen komentoja, joka ohjaa robotin eri alijärjestelmien toimia. Välikieli koostuu joukosta komentoja, jotka suoritetaan hallitusti. Komennot voivat hyödyntää mittauksia useasta anturista suorituksen aikana. Komentojen erilainen järjestäminen mahdollistaa sen, että robotti pystyy suorittamaan monimutkaisia ja yhtäaikaisia tehtäviä alustallaan ja manipulaattoreillaan (Halme et al. 2003).



Kuva 4. WorkPartnerin ohjausarkkitehtuuri.

2.4 WorkPartnerin tekniikan tila

Diplomityön tekohetkellä WorkPartnerin mekaaniset ja elektroniset rakenteet ovat valmiit lukuunottamatta manipulaattorien toista (vasenta) rannetta, jonka tilalla on tällä hetkellä ihmisen käden muotoinen liikkumaton uloke. Eniten tehtävää on ohjelmointipuolella, jossa parhaillaan tehdään ja parannellaan navigointijärjestelmää

ja kognitiivista käyttöliittymää. Robotille on ohjelmoitu vain yksi varsinainen työtehtävä aikaisemmin ja tämä liittyi porausreikien automaattiseen tunnistamiseen ja mittaamiseen (Sievilä 2003).

Tulevaisuudessa WorkPartneriin pyritään saamaan tietokanta, johon on tallennettu mahdollisia ympäristön kohteita ja niiden ominaisuuksia. Tämän avulla robotti tietäisi esim. laatikon tai pallon ominaisuuksia ja pystyisi sellaisen nähdessään tunnistamaan kyseisen kappaleen. Tietokannassa olisi yksinkertainen kolmiulotteinen malli kohteesta ja kameralla otettu kaksiulotteinen värikuva. Tätä tietokantaa on tarkoitus käyttää välipintana ihmisen ja robotin välisessä kommunikoinnissa (Suomela & Halme 2002).

2.5 Teleoperointivaljaat

Teleoperointivaljailla (kuvassa 5) pystytään ohjaamaan WorkPartnerin alustan liikkeitä etäältä. Laitteella mitataan käyttäjän ranteiden sijainti ja asento suhteessa saman puolen olkapäähän. Laitetta voidaan käyttää monella tavalla robotin ohjaamiseen. WorkPartnerin manipulaattoreita voidaan ohjata teleoperoidusti, koska käyttäjän käden liikkeet saadaan valjaiden avulla siirrettyä robotin manipulaattorien liikkeiksi. Myös WorkPartnerin alustaa voidaan ohjata teleoperoidusti samalla laitteella. Teleoperointivaljaisiin pukeutuneen käyttäjän käsissä on kapulat, jotka sisältävät kiihtyvyysanturit käyttäjän ranteen asennon mittaamiseksi. Kapuloissa on myös liipaisimet ja rullasäätimet, joilla voidaan muun muassa ohjata manipulaattorien tarttumista ja otteessa käytettävää voimaa. Teleoperointivaljaita on leikillisesti kutsuttu myös yo-yo-ohjaimeksi, koska käsien paikkaa mittaavat vaijerikelat muistuttavat jojoa.

Kuva 5. Teleoperointivaljaat käytössä (Kivi A.).

3 Oppiminen

3.1 Oppimisen taustaa

Kaikesta kehittymisestään huolimatta roboteilla on vaikeuksia toimia arvaamattomassa ympäristössä. Jotta robotti pystyisi toimimaan kuin eläin, on sen kyettävä ajattelemaan ja kehittymään jossain määrin itsenäisesti. Eläimen kaltainen ajattelu on aika vaikea tavoite robotille, mutta entä kehityskyky? Jotta robotti kehittyisi eli tekisi jotain paremmin kuin ennen, on sen pystyttävä oppimaan virheistään ja pystyttävä kehittymään sekä fyysisesti (robotin rakenne) että henkisesti (robotin ohjelmisto).

Robotin fyysinen kehittyminen voidaan tällä hetkellä saada aikaan vain valmistamalla edeltäjiään parempia robotteja. Yleisimmin tämä tarkoittaa sitä, että tehdään uusi robotti kutakin erikoistehtävää varten. Toisaalta on myös luotu jo robotteja, jotka pystyvät muuntautumaan eri tehtäviä varten toimintansa aikana. Nämä robotit pystyvät muuttamaan rakennettaan muuttamalla ulkomuotoaan tai vaihtelemalla käytettäviä manipulaattoreita (työkaluja). Muuntautumiskykyiset robotit ovat kuitenkin vasta kehitysasteella, mitään käytännön sovelluksia ei ole tehty. Yksi esimerkki tämänlaisen robotin rakenteesta on esitetty Farritorin ja Dubowskyn työssä (Farritor et al. 2001).

Robotin "henkinen" kehittyminen on fyysistä helpompi toteuttaa, koska muuntautumista eri tehtävien suorittamista varten ei tarvitse muuttaa laitteen rakennetta. Jotta robotti voisi kehittyä "henkisesti", on sen osattava suorittaa tehtävä jollakin tasolla (huonollakin) ja kyettävä omatoimisesti parantamaan tehtävän suorituksen tasoa. Eli sen on siis osattava oppia.

3.2 Mitä on oppiminen?

Yksi määritelmä oppimiselle on käyttäytymisen muuttuminen yksilön kokemuksen perusteella. Tämä tarkoittaa sitä, että jos organismi pystyy havaitsemaan ja muuttamaan käyttäytymistään, voidaan sanoa, että se kykenee oppimaan (Encyclopædia Britanica 2003). Toisen määritelmän mukaan oppiminen tuottaa muutoksia agentissa, joka mahdollistaa, että ajan kuluessa se kykenee toimimaan tehokkaammin ympäristössään (Ogino et al. 2003).

Behavioristisessa oppimiskäsityksessä sanotaan, että oppiminen perustuu palautteisiin, joita oppija saa reagoidessaan ympäristöönsä. Oppija toistaa niitä toimintoja, joihin liittyy mielihyvää ja välttää niitä toimintoja, joihin taas liittyy epämiellyttäviä elämyksiä (Sinkkonen 2002).

3.3 Eläimen oppiminen

Oppiminen on eläimille helppoa ja luonnollista. Eläimet oppivat joka hetki jotakin, sekä hyviä että huonoja asioita. Robotilta tämä ei käy niin helposti. Robotin, kuten WorkPartnerin, pitää tietää mitä se tavoittelee. Robotin pitää tietää, minkälainen suoritus on "huono" ja minkälainen suoritus on "parempi" kuin aiempi yritys ja milloin tavoite on saavutettu eli suoritus on "riittävän hyvä". Robotti ei ole tietoinen ympäristöstään samalla tavalla kuin ihminen on ja robotin on hankala ymmärtää itse, mitä se on oppimassa.

Eläimillä pääasiallinen oppimiseen motivoiva tekijä on selviytyminen. Eläimet tekevät suurimman osan päätöksistään selviytymisen kannalta. Ne parantavat tehtävän (esim. "hae pähkinöitä puusta") suoritusta yrittämällä tehdä sen mahdollisimman nopeasti ja tehokkaasti. Samalla ne tutkivat uusia keinoja saada pähkinä puusta alas. Elävät olennot ovat kehittyneet erilaisissa ympäristöissä. Ne muodostavat ekosysteemejä sopeutumalla muuttuvaan ympäristöönsä, etsimällä paikkaa, jossa elää ja muokkaamalla ympäristöään. Olennot käyttävät geneettisesti niihin ohjelmoitua tietoa selviytyäkseen ja samalla keräävät tietoa mukautuessaan ympäristöönsä (Fukuda et al. 1997).

Eläimillä on paljon geneettisesti koodattua tietoa, asioita joita ne osaavat vaiston varaisesti. Tämä tieto ei kuitenkaan riitä selviytymiseen, joten eläimen on saatava tietoa myös muualta. Eläimet oppivat pääasiassa matkimalla. Ne matkivat enimmäkseen omia lajitovereitaan, mutta myös muita eläimiä. Eläimet oppivat myös vihollisiltaan ja saaliiltaan. Juostessaan karkuun tai metsästäessään eläimet seuraavat toisiaan ja yrittävät aavistaa, miten toinen reagoi. Tarkkailemalla jatkuvasti ympäristöään ne pystyvät ymmärtämään luonnon näyttämät pienimmätkin merkit ja esimerkiksi aavistavat vaaran uhkaavan niitä.

Eläimet käsittelevät tietoa hieman eri tavalla kuin robotit. Mobiili robotti liikkuessaan tallentaa yleensä paikkansa suhteessa ympäristöönsä koordinaattien avulla. Eläimet taas tunnistavat paikkansa aistimista tulevan tiedon perusteella. Eläin muistaa jonkin

paikan, koska se tunnistaa sen näön, kuulon ja hajun perusteella. Robotti taas liikkuessaan laskee paikassa tapahtuvan muutoksen ja muuttaa koordinaatistoa sen perusteella, ei sen mitä se havaitsee. Jotta robotti saataisiin toimimaan samalla tavalla kuin eläin, on sen pyrittävä toimimaan enemmän "vaiston varassa". Robotin pitäisi jättää koordinaattien käyttö liikkumisessa vähemmälle, jolloin se kokeilisi enemmän, mikä suoritus on hyvä ja oppisi, minkälainen liike vastaa koordinaateissa tapahtuvaa liikettä ("kun kättä liikuttaa näin, se vastaa koordinaatistossa yhtä metriä x-suuntaan").

3.4 Koneoppiminen

Ei liene periaatteellista estettä sille, etteikö konekin pystyisi järkeilemään monipuolisella arkitiedolla. Ongelma on lähinnä se, että maailmaa käsittelevää tietoa, käsitteitä ja tilanteita on tavattoman paljon. Esimerkiksi puhutussa kielessä on valtavasti sanoja ja niitä yhdistelemällä saatuja monimutkaisia merkityksiä. Sanat sisältävät todella paljon piilotietoa ja kokemusta maailmasta, jota ihminen oppii elämänsä aikana hallitsemaan ja jota hän käyttää lähestulkoon tiedostamatta. Lisäksi puhuessaan ihminen tuo painotuksilla ja eleillä sanoihin vielä lisää merkityksiä. Koneelle tämän kaiken opettaminen olisi hyvin työläs tehtävä, ja jos ihmisillä on usein vaikeuksia ymmärtää toisia ihmisiä, kuinka kone pystyisi siihen?

Koneella ei ole ongelmia käsitellä suurta määrää tietoa. Koneen ongelmia sen sijaan ovat tiedon hankinta (knowledge acquisition), tiedon järjestäminen todellisuutta vastaavaksi ja toimivaksi maailmankuvaksi, sopivan esityskielen löytäminen sekä tiedon ja sen käytön hallinta.

Yleisimmin tietokone osaa päätellä oikein vain yksinkertaisissa ja samanmuotoisissa tilanteissa ja käyttäen vain tietyn tyyppistä päättelyä kerrallaan. Kone ei kykene hallitsemaan ja joustavasti yhdistelemään erilaisia tiedon ja päättelyn muotoja. Kone töksähtää helposti umpikujaan tai joutuu harhateille (esim. ikuiseen silmukkaan) eikä osaa palata ja valita uutta järkevää suuntaa. Eksyksissä se voi kuluttaa loputtomasti aikaa löytämättä mitään järkevää ratkaisua. Kone ei osaa "nähdä metsää puilta". Sen on vaikeaa havainnoida ja arvioida sen ympärillä olevia tilanteita, ja keksiä näihin uudentyyppisiä ratkaisuja (Seppänen 1990).

Periaatteessa koneen oppimiselle ei näytä olevan esteitä. Kone ei siten ole ihmistä tai eläintä huonommassa asemassa. Jos otetaan huomioon koneen nopeus ja luotettavuus,

voidaan olettaa, että sen edellytykset oppimiseen ovat jopa eläintä paremmat. Kone ei tarvitse yhtä monimutkaista fysiologiaa oppimisen tueksi. Sen "henkiset" kyvyt eli ohjelmat (software), ovat sen "ruumiista" eli laitteistosta (hardware), riippumattomia. Jokin valmiiksi opittu asia voidaan tallentaa koneen muistiin ja siirtää toiseen koneeseen paljon helpommin kuin eläimet pystyvät siirtämään tietoa toiselle eläimelle. Toisaalta se, että eläinten oppimisen mekanismit ovat hyvin moninaisia, saattaa olla tärkeä syy siihen miksi, eläimen oppiminen on niin menestyksellistä.

3.5 Robottien oppiminen

Viimeistään elokuvat, muun muassa Tähtien sota –elokuvat, ovat tuoneet robotit suuren yleisön tietoisuuteen. Tänä päivänä eniten robotteja voi löytää tehdassaleissa ahertamassa ja tutkijoiden kammioissa. Nämä robotit ovat kuitenkin vielä kaukana avaruuselokuvien antamasta kuvasta, jossa robotti on ihmisen mekaaninen kopio ja osaa toimia lähestulkoon yhtä älykkäästi kuin ihminenkin. Todellisuudessa tavallinen robotti on usein vain tietokoneen ohjaama mekaaninen käsivarsi, joka liikkuu servomoottoreiden ohjaamana ja jonka päässä on työkalu, vaikkapa hitsauspilli. Tutkijoiden laboratorioista löytää kuitenkin kehittyneempiä robotteja, jotka osaavat ympäristön havaintojen perusteella tehdä yksinkertaisia päätöksiä ja vieläpä parantaa tehtävänsuoritustaan kokeilemalla erilaisia ratkaisuja. Nämä oppivat robotit ovat kuitenkin usein prototyyppiasteella ja kestänee vielä vuosikymmeniä ennen kuin näitä robotteja nähdään markkinoilla. Hyvää tietoa oppimisesta robotiikassa löytyy Nolfin ja Floreanon (1999) kirjoittamasta "Learning and Evolution" tieteellisestä artikkelista.

Robotti, joka pystyisi oppimaan koko ajan kaikesta, olisi hyvin arvaamaton (niinkuin ihmisetkin ovat). Ja koska ihmiset eivät vielä ymmärrä kaikkia ihmisten käytökseen vaikuttavia asioita, saattaisi olla vaarallista yrittää tehdä täysin ihmisen kaltaista robottia. Toisaalta robotti, kuten WorkPartner, jolle annetaan yksittäisiä oppimistehtäviä, on huomattavasti turvallisempi ja helpommin ohjattavissa. Kun siis ihminen haluaa robotin oppivan jonkin tehtävän, hän asettaa robotin oppimistilaan ja rupeaa opettamaan robottia tehtävän suorittamisessa. Kun robotti on käyttäjän mielestä oppinut tehtävän suorituksen tarpeeksi hyvin, ihminen käskee robotin lopettamaan oppimisen. Tällöin robotti opiskelee uusia asioita vain hallituissa ympäristöissä. Muuna aikana robotti vain toistaa jo opittuja asioita.

Teollisuusrobotin ei tavallisesti tarvitse tarkkailla ympäristöään. Robotti siirtää piisirut aina saman kaavan mukaan paikasta toiseen. Monimutkaisempaa työtä tekevän robotin täytyy kuitenkin saada tietoa ympäristöstään. Liikkuvan robotin täytyy nähdä eteensä, jotta se osaisi väistää vastaan tulevat esteet. Ympäristöään muokkaavan robotin täytyy voida osata erottaa erilaiset objektit toisistaan. Esim. WorkPartnerin on tunnistettava laatikko, jonka se aikoo nostaa.

Monet tehtävät, jotka ovat robotille vaikeita suorittaa, ovat ihmiselle huomattavasti helpompia. Ihminen osaa yleensä helposti erottaa vierekkäin pysäköidyn urheiluauton ja pakettiauton toisistaan. Robotin tietokoneelle saman tehtävän suoritus vaatii monimutkaisia laskutoimituksia ja yksityiskohtien vertailuja. Lisäksi ihmisellä on hallussaan paljon kokemusperäistä tietoa, jota on vaikea saada siirrettyä robotille. Tavallinen ihmisautoilija kykenee sanomaan, kuinka tulee peruuttaa ja kääntää ohjauspyörää, jotta hän saisi autonsa liikkumaan ahtaassa pysäköintihallissa. Jos Robotti yrittää samaa, sen täytyy jatkuvasti laskea liikuteltavan auton koko ja muoto, jotta se ei kolhisi viereiseen ruutuun pysäköidyn toisen auton kanssa. Sen pitäisi myös laskea etukäteen, missä järjestyksessä autoa tulee ajaa eteen- ja taaksepäin (Eklund 1990).

3.6 Oppimisen eri muotoja

Tekoälytutkimuksessa on jo pitkään pyritty saamaan koneita toimimaan ja oppimaan kuin luonnossa elävät eläimet. Tästä tutkimuksesta on syntynyt eri oppimisen muotoja, jotka enemmän tai vähemmän muistuttavat eläinten oppimista. Alla on lueteltu yleisimpiä oppimisen muotoja:

- Neuroverkot (Neural networks)
- Evolutiivinen oppiminen (Evolutionary learning)
- Oppiminen esimerkistä (Learning from experience)
- Induktiivinen oppiminen (Inductive learning)
- Selityspohjainen oppiminen (Explanation-based learning)
- Monistrategiaoppiminen (Multistrategy learning)

Oppimisessa saatava palaute on yleensä tärkein tekijä, joka vaikuttaa siihen, minkä tyyppistä oppimismenetelmää käytetään. Koneoppimisessa menetelmät jaetaan

yleensä kolmeen luokkaan sen mukaan, minkälaista palautetta tehtävän suoritukseen on saatavilla. Menetelmät on esitetty alla:

- Ohjattu oppiminen (supervised learning)

  Jos annettava palaute, joka tulee opettajalta tai ympäristöstä, on oikein, kutsutaan oppimista ohjatuksi oppimiseksi.

- Ohjaamaton oppiminen (unsupervised learning)

  Jos saadusta palautteesta ei tiedetä, onko se "hyvää" vai "huonoa", on kyseessä ohjaamaton oppiminen.

- Vahvistava oppiminen (Reinforcement learning)

  Suorittaessaan tehtävää saadaan palautetta, joka annetaan palkinnon muodossa. Ei ole opettajaa kertomassa, mikä on oikein, vaan hyvästä suorituksesta annetaan palkinto ja oppijan pitää itse ymmärtää, mitä hän teki oikein.

Eri oppimismuotojen käyttöön vaikuttaa kolme eri asiaa: "1) Mitkä komponentit tehtävän suorituksesta ovat opittavissa. 2) Minkälaista palautetta on saatavissa näiden komponenttien oppimiseksi. 3) Minkälaista esitysmuotoa komponenteilla käytetään." (Russell 1995 kohta 18.1).

Tässä työssä on tavoitteena, että WorkPartnerille voidaan opettaa liikkeitä esimerkin avulla. Jos laatikon nosto opetetaan esimerkin avulla, opettava ihminen tallentaa manipulaattorien liikkeet laatikkoa nostettaessa joko manuaalisesti syöttämällä arvot Wopalle tai esim. teleoperointivaljaita hyödyntäen. Tällöin raajojen liikkeet ovat jo alussa melko luonnollisen näköisiä. Oppimistapahtumassa raajojen liikkeet lähestyvät jokaisen oppimissyklin jälkeen kerta kerralta "tehokkaampia" suorituksia, jolloin ihmismäisen luonnollisuuden ja sulavuuden pitäisi korostua (varsinkin jos ajattelemme ihmisten liikkeitä tehokkaiksi, mitä ne eivät välttämättä ole).

4 Geneettinen algoritmi

4.1 Geneettisen algoritmin taustaa

Geneettiset algoritmit (GA) keksi John Holland 1960 luvulla. Hän jatkokehitti algoritmeja oppilaidensa ja kolleegoiden kanssa Michiganin yliopistossa 60- ja 70-luvuilla. Toisin kuin evolutiivisissa strategioissa ja ohjelmoinnissa, Hollandin alkuperäinen tavoite ei ollut kehittää algoritmeja ratkaisemaan tiettyjä ongelmia, vaan tutkia mukautumisen ilmiötä (adaptation) sellaisena kuin se ilmenee luonnossa. Hän yritti keksiä tapoja, joilla tämä luonnon mekanismi saataisiin tuotua tietokonejärjestelmiin.

Hollandin GA on menetelmä, jolla siirrytään kromosomipopulaatiosta (jono nollia ja ykkösiä eli bittijono) toiseen käyttämällä luonnollista valintaa (natural selection) yhdessä geneettisten operaatioiden (risteytys, mutaatio ja inversio) kanssa. Jokainen kromosomi koostuu joukosta geenejä (geeni puolestaan koostuu biteistä) ja jokainen geeni on tietyn alleelin eli bitin instanssi (0 tai 1). Luonnollisessa valinnassa populaation kromosomeille annetaan hyvyys-arvo eli fitness-arvo, joka kertoo kromosomin lisääntymiskelpoisuuden. Valintaoperaatio valitsee ne kromosomit, joiden annetaan lisääntyä eli joilla on paras fitness-arvo. Tästä johtuen paremman fitness-arvon omaavilla kromosomeilla on suurempi todennäköisyys lisääntyvät muita enemmän.

Lisääntymisessä valitut kromosomit risteytetään keskenään, jonka jälkeen yksittäisissä geeneissä voi syntyä mutaatioita. Lisäksi tietyissä kromosomin pätkissä saattaa tapahtua inversio. Risteytyksessä kromosomit vaihtavat osia keskenään matkien näin biologiassa tapahtuvaa risteytystä. Mutaatio satunnaisesti muuttaa alleelien eli yksittäisten bittien arvoja päinvastaiseksi (nollasta tulee ykkönen ja ykkösestä nolla). Inversio taas kääntää kromosomin osan geenien järjestyksen päinvastaiseksi (Mitchell, M. 1996).

4.2 Geneettisen algoritmin ominaisuuksia

Geneettisissä algoritmeissa (GA) on tiettyjä erikoisia ominaisuuksia verrattuna moniin muihin optimointimenetelmiin. Yksi GA:n ominaispiirteistä on koko parametrijoukon (populaatioiden) yht'aikainen käsittely, toisin kuin monissa muissa optimointimenetelmissä, joissa yritetään optimoida vain yhtä parametria kerrallaan.

Hyviä ratkaisuja etsitään siis suurella joukolla: Vaikka yksi parametreista jäisi jumiin johonkin lokaaliin ratkaisuun, kaikki muut parametrit etsisivät vielä muita ratkaisuja. Tämä nopeuttaa etsimistä huomattavasti. GA tarvitsee vain kustannusfunktion (eli fitness-funktion) antamaa tietoa etsiessään parempia ratkaisuja. Mitään lisäfunktioita, kuten derivaattoja, ei tarvita. Tämä tekee GA:n käytöstä käytännössä varsin helppoa. GA käyttää satunnaisfunktioita osana algoritmiaan. GA ei yritä löytää ratkaisua satunnaisesti, vaan se käyttää ratkaisun etsimisessä apuna suoraa satunnaista etsintää (direct random search). Tämä antaa GA:lle vapauden toimia itsenäisesti, ongelmasta riippumatta (Halme, J. 1997).

Robottia on hyvin vaikea ohjelmoida tekemään jokin monimutkainen tehtävä. Toisaalta on paljon helpompaa kehittää kriteerejä, jotka kertovat kuinka hyvin robotti suoriutuu vaikeistakin tehtävistä. Miksi suotta yrittää saada robotti tekemään jotain vaikeaa, kun geneettisellä algoritmilla voidaan antaa robotin itse yrittää ratkaista vaikea tehtävä. Riittää kun robotille kerrotaan, että "tuo oli huono suoritus" tai "tuo meni hieman sinne päin." (Husbands et al. 1997).

GA:ssa on ominaisuuksia, joiden takia on tärkeää tasapainottaa lisääntyvien ratkaisujen valintaa. Jos valitaan vain parhaat ratkaisut kustakin sukupolvesta, syntyy hyvin suppea valikoima seuraavan vaiheen ratkaisuja, koska kaikki muut ratkaisuvaihtoehdot menetetään ja monimuotoisuus populaatiosta katoaa. Tämän jälkeen populaatiossa tapahtuu hyvin vähän muutoksia ja kehitys jämähtää paikalleen ennen aikojaan. Jos taas valitaan populaatiosta hyvin laaja joukko lisääntyviä ratkaisuja (esim. asettamalla ratkaisujen fitness-arvot liian lähelle toisiaan), ratkaisut kehittyvät hyvin hitaasti. Tällöin saadaan paljon erilaisia ratkaisuja, mutta koska lähes kaikki saavat lisääntyä samalla nopeudella, 'hyvien' ratkaisujen joukko ei muodostu 'huonoja' suuremmaksi (Mitchell, M. 1996).

4.3 Työssä käytetty menetelmä

Tässä diplomityössä käytetään niin sanottua rulettipyörävalintaa (Roulette wheel selection). Tässä valintamenetelmässä jokainen populaation kromosomi eli ratkaisu sijoitetaan pyörän osaksi siten, että pyörä jaetaan yhtä moneen osaan kuin populaatiossa on ratkaisuja ja osien koot riippuvat ratkaisujen fitness-arvoista. Jos ratkaisulla on pieni fitness-arvo, se saa rulettipyörästä pienen sektorin ja jos fitness-

arvo on suuri, on myös sektori pyörässä suuri. Ratkaisun fitness-arvon suuruus on suoraan verrannollinen rulettipyörän sektorin suuruuteen.

Tässä valintamenetelmässä rulettipyörää pyöräytetään populaation koon verran kertoja, jonka jälkeen ruletissa valikoituneet ratkaisut saavat lisääntyä keskenään. Ratkaisuista muodostetaan parit, joihin käytetään risteytysoperaatiota. Risteytyminen tapahtuu tyypillisesti todennäköisyydellä 0.6 - 0.95. Jos risteytys tapahtuu, arvotaan kromosomista kohta, mihin se tehdään. Tällöin saadaan kaksi uutta ratkaisua, joissa on geenejä kummastakin kromosomista. Jos risteytymistä ei tapahdu, kopioidaan ratkaisut sellaisenaan jatkoon. Risteytymisen jälkeen ratkaisuissa voi tapahtua myös mutaatioita. Jokaisen ratkaisun bitille tehdään tarkistus, muuttuuko se vai ei. Mutaation todennäköisyys bittiä kohden on yleensä noin 0.001 – 0.01. Tässä menetelmässä ratkaisuille ei tehdä inversio-operaatioita (Mitchell, M. 1996).

Rulettipyörävalintamenetelmä valittiin pääasiassa siksi, että se on hyvin suosittu ja siis myös hyvin testattu menetelmä. Menetelmässä ei hylätä huonoja ratkaisuja suoraan. Huonoillekin ratkaisuille annetaan mahdollisuus (tosin pieni) lisääntyä, jolloin huono ratkaisu risteytyessään hyvän kanssa voi tuottaa selkeästi tavanomaisesta poikkeavia suorituksia ja näin voi syntyä innovatiivisiakin ratkaisuja ongelmiin. Jos ratkaisu kuitenkin halutaan hylätä jostain syystä, voidaan sille antaa fitness-arvoksi 0, jolloin sillä ei ole paikkaa rulettipyörässä.

Rulettipyörävalinnassa voi esiintyä ongelmia, kun eri ratkaisujen fitness-arvot eroavat toisistaan hyvin paljon. Tällöin ratkaisut, joilla on suuri fitness-arvo, valitaan lähestulkoon aina. Toisaalta ratkaisuilla, joiden fitness-arvo on hyvin pieni, on hyvin pieni mahdollisuus tulla valituksi. Tämä ongelma korostuu entisestään kun populaatioiden koko on melko pieni (noin 10).

Tämä ongelma on toisaalta myös vahvuus, jos käyttäjä itse antaa fitness-arvon ratkaisuille. Tällöin käyttäjä pystyy antamiensa fitness-arvojen eroilla vaikuttamaan selkeästi geneettisen algoritmin toimintaan. Jos käyttäjä haluaa paljon vaihtelua ratkaisuihin, hän antaa hyvän mahdollisuuden kaikille ratkaisuille valitsemalla fitness-arvot läheltä toisiaan. Toisaalta, jos käyttäjä haluaa hyvin samanlaisia ratkaisuja kuin parhaat ratkaisut sillä hetkellä, voi hän asettaa fitness-arvoille suuret erot. Tällä tavalla käyttäjä voi säätää geneettisen algoritmin käyttäytymistä oppimisen aikana sen mukaan, mihin suuntaan hän haluaa robotin toiminnan suuntautuvan. Tällä tavalla

rulettipyörävalinta saadaan muistuttamaan muitakin menetelmiä, kuten arvovalintaa (rank selection) ja sopivilla valinnoilla myös Boltzmannin valintaa (Boltzmann selection). Nämä menetelmät esitellään hieman myöhemmin (katso kohta 4.5).

4.4 Geneettisen algoritmin parametrit

GA:ta käytettäessä on päätettävä, miten sen parametrien arvot, kuten populaation koko sekä risteytyksen ja mutaation todennäköisyys, valitaan. Ei ole tehty tyhjentävää tutkimusta siitä, mitkä olisivat parhaat arvot parametreille. Usein käytetään niitä arvoja, jotka ovat toimineet muilla. Schaffer, Caruana, Eshelman ja Das (1989) tekivät tutkimuksen, jossa he järjestelmällisesti testasivat suuren määrän erilaisia parametrikombinaatioita ja niiden vaikutusta toisiinsa. He huomasivat että ratkaistavalla ongelmalla ei ollut vaikutusta siihen, mitkä ovat parhaat valinnat populaation koolle ja risteytyksen ja mutaation todennäköisyydelle. Nämä arvot olivat populaation koolle 20 – 30, risteytyksen todennäköisyydelle 0,75 – 0,95 ja mutaation todennäköisyydelle 0,005 – 0,01 (Schaffer et al. 1989). Tässä diplomityössä käytettiin arvoina populaation kokoa 10, risteytyksen todennäköisyyttä 0,8 ja mutaation todennäköisyyttä 0,01.

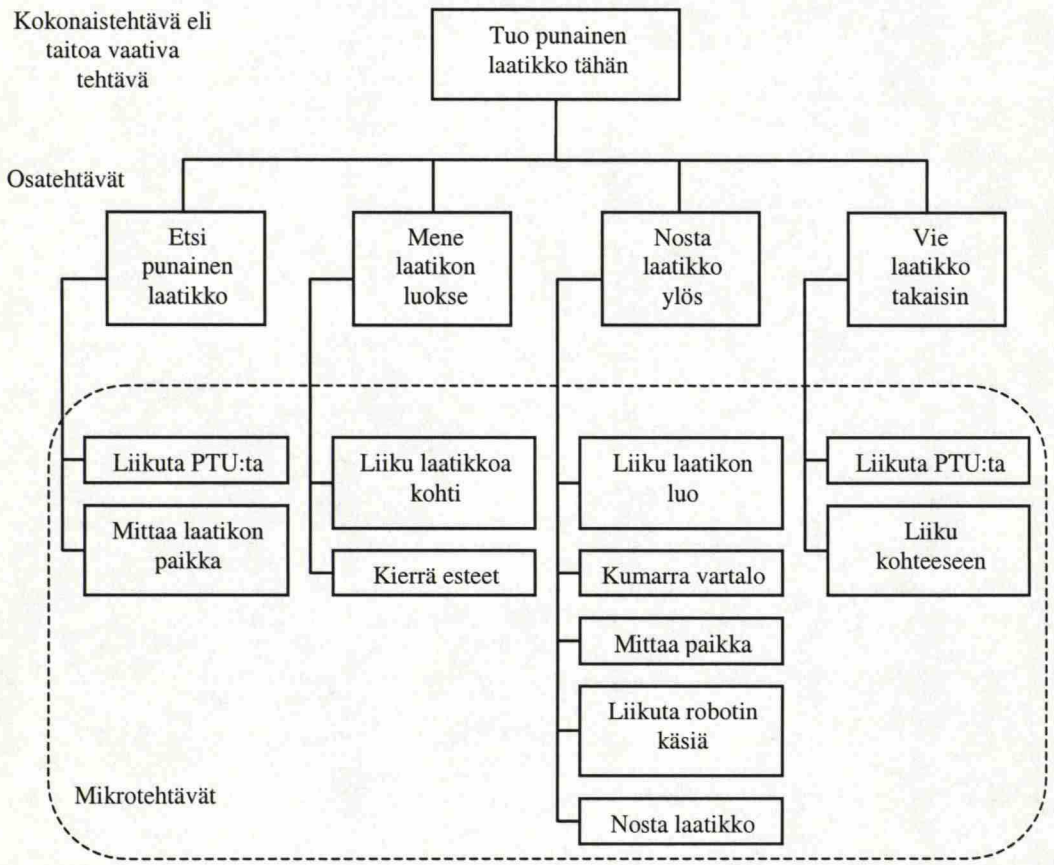
4.5 Muita valintamenetelmiä

Muita menetelmiä, joilla oppimisen olisi voinut suorittaa, ovat "elitismi" (Elitism), Boltzmannin valinta (Boltzmann selection) ja arvovalinta (Rank selection). "Elitismi" (De Jong 1975) on lisäys moniin muihin valintamenetelmiin. Se pakottaa GA:n säilyttämään osan parhaista ratkaisuista populaatiossa, jos ne olisivat häviämässä sen takia että niitä ei sattumalta valittu lisääntymään tai jos risteytys tai mutaatio olisi tuhoamassa ne. Boltzmannin valinnassa (Goldberg 1990) lisääntymisen painetta muutetaan ajan kuluessa. Esimerkiksi aluksi sukupolven vaihtuessa saavat lähes kaikki ratkaisut lisääntyä, jolloin saadaan paljon variaatiota ratkaisuihin. Sukupolvien kuluessa lisääntymiskriteerejä hiljalleen tiukennettaan, jolloin vain parhaimmat ratkaisut saavat lisääntyä. Arvovalinnassa (Baker 1985) fitness-arvot annetaan ratkaisuille porrastetusti. Huonoin ratkaisu saa arvon 1, seuraava arvon 2, jne. Valinta tapahtuu kuin rulettipyörävalinnassa, mutta rulettipyörään sijoitetaan ratkaisujen arvot eikä fitnessiä. Tällä menetelmällä estetään liian nopeaa erikoistumista tietyn tyyppisiin ratkaisuihin.

5 Laatikon etsintä- ja nostotehtävä

5.1 Tavoitteet

Tavoitteena laatikon etsintä- ja nostotehtävässä on, että WorkPartnerille voitaisiin antaa taitoa vaativa tehtävä, jossa robottia voitaisiin käskyttää yksinkertaisesti komennolla "Tuo punainen laatikko tähän," tai englanniksi sanottuna "Bring red box here." Kokonaistehtävässä "tuo punainen laatikko tähän" on neljä osatehtävää: "etsi punainen laatikko", "mene laatikon luokse", "nosta laatikko ylös" ja "vie laatikko annettuun paikkaan". Osatehtävissä on useita mikrotehtäviä, jotka ovat pieniä toimia, kuten etäisyyden mittaus laserilla, robottialustan liike tai manipulaattorin siirto määriteltyyn paikkaa. Alla olevassa kuvassa 6 on esitetty tämän taitoa vaativan tehtävän hierarkinen rakenne. Ylimmällä tasolla on kokonaistehtävä, keskimmäisellä tasolla ovat osatehtävät ja alimmalla tasolla sijaitsevat mikrotehtävät (kuvassa kehystettyinä katkoviivalla).



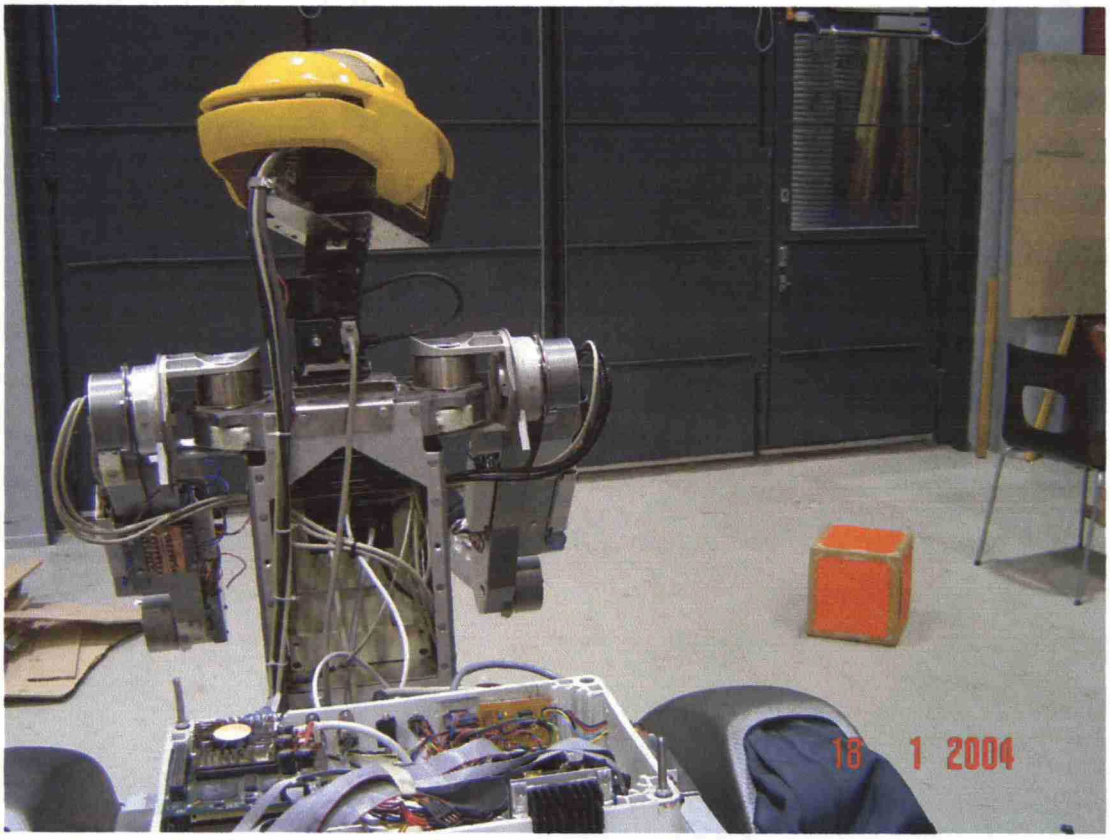
Kuva 6. Taitoa vaativan tehtävän hierarkinen rakenne.

Tässä työssä on keskitytty laatikon etsintään ja laatikon nostoon. "Laatikon luokse liikkuminen" on toteutettu yksinkertaisena siirtymisenä ja "johonkin paikkaan liikkuminen" on jätetty kokonaan pois. Tämä siksi, että geneettisen algoritmin soveltamisessa etsintä- ja nostotehtävät ovat kaikkein haastavimpia ja niissä päästään hyvin soveltamaan geneettisen algoritmin käyttöä eri tilanteisiin. Liikkuminen paikasta toiseen on yksinkertaisempi ongelma, jonka hyvään suorittamiseen ei välttämättä tarvita geneettistä algoritmia lainkaan. Lisäksi geneettisen algoritmin käytöstä liikkumisessa on jo lukuisia tutkimuksia. Tutkimuksia on tehty esimerkiksi mobiilin robotin oppimisesta ja adaptaatiosta ulkona (Hagras et al. 2001), mobiilin robotin oppimisesta havaintopohjaisella geneettisellä algoritmilla (Kobuta et al. 2000) ja mobiilin robotin oppimisesta ja adaptaatiosta maanviljelyksen ylläpitämiseen (Hagras et al. 2002).

5.2 Osatehtävien kuvaus

5.2.1 Laatikon etsintä

"Laatikon etsintä" –tehtävässä on tavoitteena löytää tietty laatikko (tässä punainen laatikko). WorkPartner etsii kääntelemällä päätään eli PTU-yksikköä puolelta toiselle sekä ylös ja alaspäin. Se tunnistaa laatikon kuvankäsittelyn avulla eli etsii punaisia laatikon muotoisia kappaleita (katso kuva 7). Tunnistettuaan laatikon se mittaa etäisyyden ja paikan laser-etäisyysmittarilla. Tarkoitus on että kun geneettistä algoritmia sovelletaan tähän tehtävään, WorkPartner pikkuhiljaa nostaa laatikon etsinnän nopeutta siten, että algoritmi oppii katsomaan ensin ne paikat, missä laatikko todennäköisimmin on, kuten "lattialla jossain", eikä lähde esimerkiksi etsimään laatikko ensin katon rajasta.



Kuva 7. WorkPartner etsii punaista laatikko ympäristöstään.

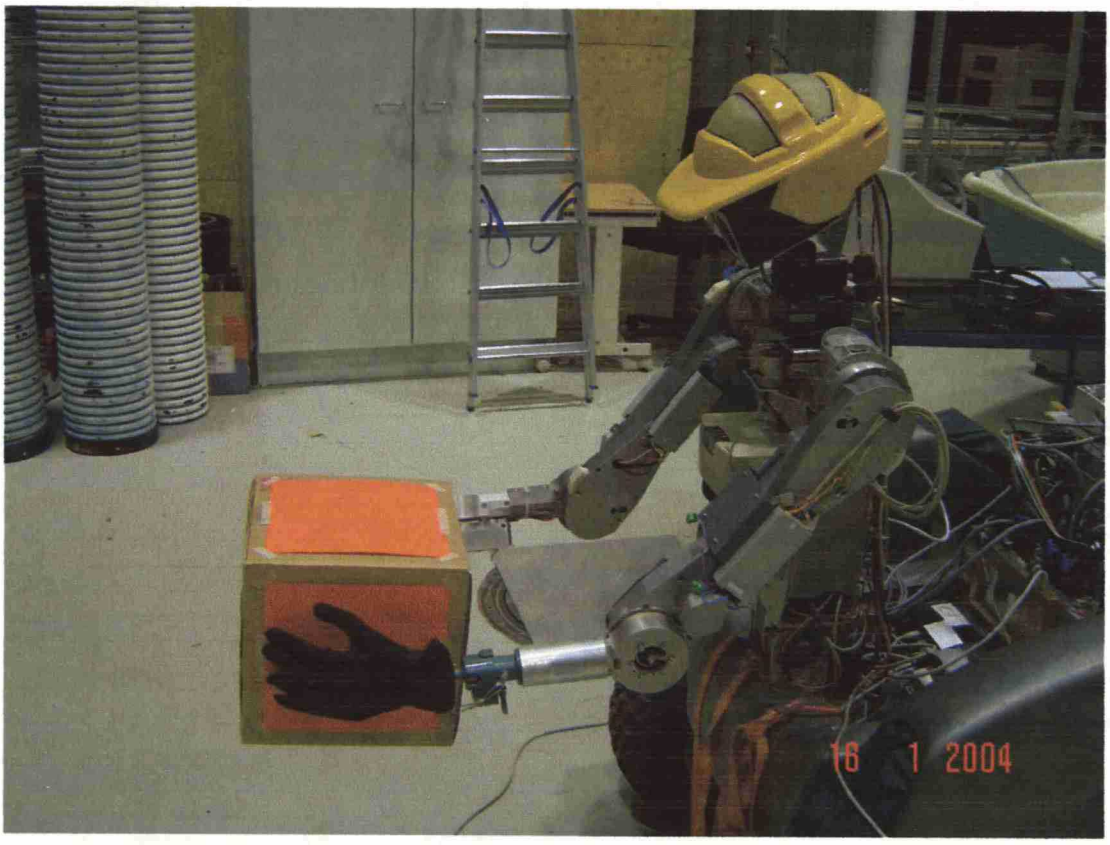
5.2.2 Laatikon luokse liikkuminen

Tässä tehtävässä WorkPartner pyrkii liikkumaan tarpeeksi lähelle laatikkoa nostaakseen laatikon ilmaan. Kun robotti on löytänyt laatikon, se pitää katseensa laatikossa ja kääntää runkoaan kulkemaan kohti laatikkoa. Laatikon luo se pyrkii ajamaan niin, että saavuttuaan laatikon eteen se on parhaassa asennossa nostamista varten (eli laatikon yksi sivu on kohtisuorassa robottiin nähden). Tehtävässä on tavoitteena, että WorkPartner havaitsee myös liikkumisesteet ja kiertää ne, mikäli mahdollista.

Tätä osatehtävää on yksinkertaistettu siten, että WorkPartner pyrkii ajamaan vain suoraan kohti laatikkoa, eikä väistele mahdollisia eteen tulevia esteitä. Tämä on tehty sen takia, että pystyttäisiin keskittymään laatikon noston oppimiseen. Tällöin ei mene aikaa siihen, että WorkPartner oppisi onnistuneesti navigoimaan laatikon luo. Esteiden väistely on kuitenkin tärkeä osa kokonaistehtävää ja oiva kohde jatkokehittelylle.

5.2.3 Laatikon nosto

"Laatikon nosto" –tehtävässä tavoitteena on saada WorkPartner ottamaan ote laatikosta, nostamaan se ilmaan ja pitämään laatikko ilmassa. WorkPartner saa otteen laatikosta asettamalla kaksi manipulaattoriaan (eli kättään) molemmin puolin laatikkoa ja liikuttamalla käsiä laatikkoa kohti. WorkPartner puristaa laatikkoa hieman, jolloin kitka pitää laatikon käsien välissä. WorkPartner vie kätensä laatikon luo tiettyä rataa pitkin ja tätä rataa muokkaamalla geneettisen algoritmin avulla saadaan aikaiseksi oppimista, jonka tuloksena laatikon noston suorituksen pitäisi nopeutua ja onnistua useammin. Robotin käsien liikkeet lienevät aluksi karkeita, mutta oppimisen kuluessa ne luultavasti tulevat pyöreämmiksi, tehokkaammiksi ja näyttävät ihmismäisemmiltä liikkeiltä (varsinkin jos oletamme että ihmisen liikkeet ovat oppimisen ansiosta tehokkaita). Kuvassa 8 on esitys testauksesta, kuinka laatikon nosto onnistuisi.



Kuva 8. WorkPartner nostaa laatikon ilmaan (testi).

Tehtävässä robotin pitää pystyä mukautumaan ympäristön tilanteeseen. Laatikon paikka ei ole aina sama ja se pitäisi pystyä nostamaan useammastakin kuin yhdestä suunnasta. Tässä osatehtävässä laatikon paikka mitataan läheltä laser-etäisyysmittarin

Kuva 8. WorkPartner nostaa laatikon ilmaan (testi).

avulla, jolloin saadaan tarkka tieto suunnasta, johon robotti käsillään kouraisee. Jos laatikko on liian alhaalla, robotti saa kätensä lähemmäksi maata kumartumalla eteenpäin. Myös tätä kumartumiskulmaa säädetään geneettisellä algoritmilla sopivaksi. "Laatikon nosto" –tehtävässä voi tulla tarve mitata laatikon paikka uudelleen, joten sekin voidaan tehdä tarpeen vaatiessa.

Mikrotehtävien järjestystä on mahdollista muuttaa, koska ennakolta ei ole varmuutta osatehtävän suorittamisen kannalta parhaasta järjestyksestä. Järjestyksen muutoksia kuvataan tilasiirtymäjonolla. Se on jono numeroita, joka kertoo, missä järjestyksessä mikrotehtävät suoritetaan. Niiden järjestystä muuttamalla saadaan robotin toiminnassa aikaan muutoksia. Tilasiirtymäjonoista kerrotaan tarkemmin kohdissa 7.3 ja 7.4.

Tästä tehtävästä on nyt jätetty pois laatikon muodon tunnistus. Tavoitteena on, että laatikon koolla ja muodolla ei ole väliä (tiettyyn rajaan asti) eli laatikon ei tarvitse olla vakiokokoa. Laatikon koko voidaan tunnistaa kuvankäsittelyn avulla tai mittaamalla manipulaattorien "tuntoaistilla" eli kun laatikkoa puristetaan ja manipulaattoreissa tuntuu vastustusta, lasketaan laatikon koko manipulaattorien asennoista.

5.2.4 Laatikon vienti kohteeseen

"Laatikon vienti kohteeseen" -osatehtävän tavoitteena on WorkPartnerin liikkuminen käyttäjän näyttämän kohteen luo, johon laatikko on tarkoitus laskea (esim. maahan). Myös tässä osatehtävässä, niin kuin "laatikon luokse liikkumisessa", pitää liikkua tietyn kohteen luo ja väistellä mahdollisia esteitä. Tässä tehtävässä voidaan tarvita myös "laatikon etsintä" –tehtävän kaltaista toimintaa, jos robotilla ei ole tiedossa, missä kohde on ja se joutuu etsimään sitä. Lopuksi tarvitaan myös "laatikon nosto" –tehtävän tyypillisiä käden liikkeitä laatikkoa laskettaessa kohteeseen. Laatikon laskeminen on kuitenkin huomattavasti helpompaa kuin nostaminen.

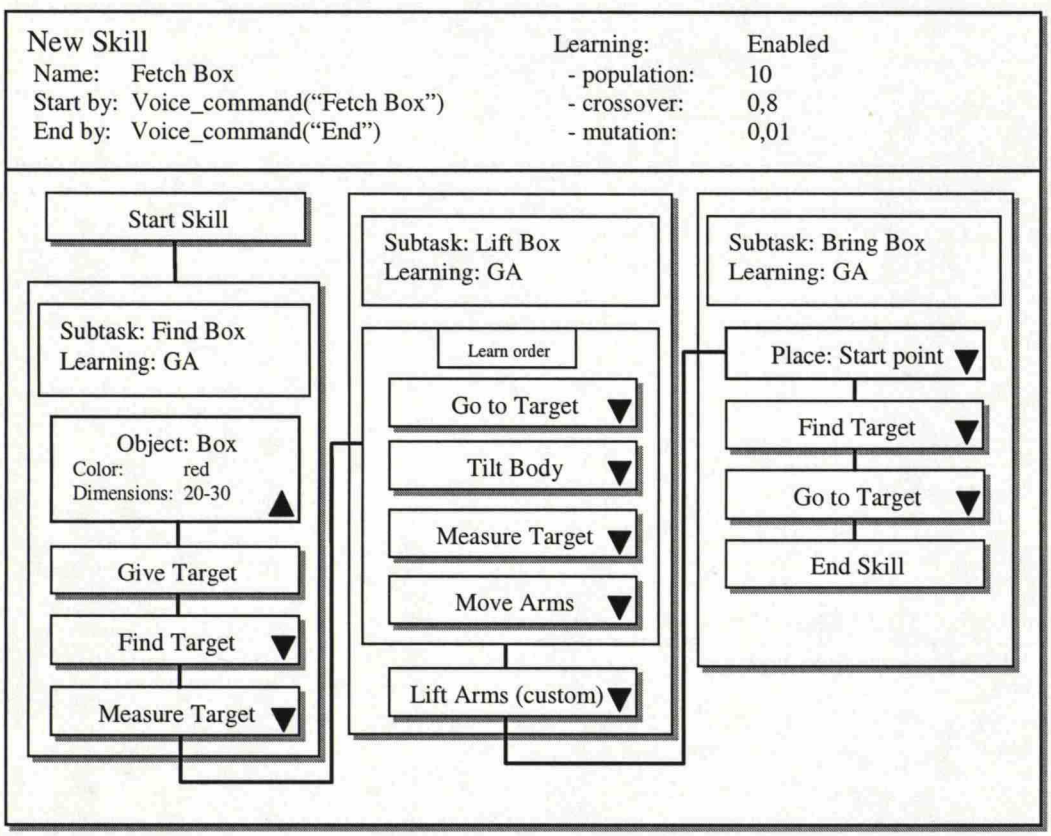
"Laatikon vienti kohteeseen" –osatehtävään ei ole tässä työssä paneuduttu, koska siinä on samanlaisia piirteitä kuin muissa osatehtävissä. Jos aikaisemmat tehtävät pystytään suorittamaan onnistuneesti, ei ole suuria ongelmia suorittaa tätäkään tehtävää. "Laatikon vienti kohteeseen" -osatehtävä on oikeastaan yksinkertaistettu yhdistelmä kolmesta muusta aiemmin mainitusta tehtävästä ja siinä mielessä hieman triviaali.

5.3 Tehtävän antaminen robotille

WorkPartnerin on saatava tallennettua tehtävä muistiinsa ennen kuin se voi alkaa opetella sitä. Taitoa vaativan tehtävän antamisen on oltava helppoa kuitenkin niin, että robotti ymmärtää tehtävän kaikki piirteet. Tehtävän annossa ei saa myöskään kadota tietoa. Taitoa vaativa tehtävä annetaan WorkPartnerille joko käyttöliittymällä tietokonepäätteen kautta tai näyttämällä mallisuoritus.

5.3.1 Tehtävänanto käyttöliittymällä

Käyttöliittymä voi olla puhtaasti tekstipohjainen, jossa pseudokoodikielellä kirjoitetaan, miten robotin tulee toimia. Tämä on kuitenkin käyttäjän kannalta huono ratkaisu, koska häneltä vaaditaan hyvää ohjelmointikykyä tehtävän syöttämiseksi robotille. Käyttöliittymä voi olla myös graafinen. Tällöin siinä yhdistelemällä graafisia komponentteja luodaan kuva, jonka käyttäjä osaa tulkita helposti ja jonka kone kääntää robotille ymmärrettävään muotoon. Graafinen käyttöliittymä lienee helpompi käyttää kuin tekstipohjainen. Kuvassa 9 on esimerkki mahdollisesta graafisesta käyttöliittymästä.



Kuva 9. Esimerkki graafisesta käyttöliittymästä robotin oppimisen ohjelmoimiseksi.

Kuvan 9 esimerkkikäyttöliittymässä aloitetaan valitsemalla ensin uusi taitoa vaativa tehtävä (new skill), jolloin ruudulle aukeaa kuvan yläreunan näkymä. Näkymässä valitaan uuden tehtävän ominaisuuksia kuten nimi, miten tehtävä käynnistetään ja suljetaan, onko oppiminen aktivoitu sekä oppimisen parametreja. Seuraavaksi käyttöliittymässä aletaan koota tehtävän suoritukseen tarvittavia palasia. Käyttöliittymän työpöydälle kerätään mikrotehtäviä (kuvassa esim. "Find Target" ja "Move Arms") ja ne jaotellaan osatehtäväjoukkoihin. Osatehtävissä mikrotehtävät liittyvät oleellisesti toisiinsa ja niillä on samoja ominaisuuksia (kuvassa "Find Box"). Jos käyttöliittymässä ei ole valmiina tarvittavaa mikrotehtävää, voi sen rakentaa itse esimerkiksi liittämällä alemman tason liikekomentoja peräkkäin tai muokkaamalla jo olemassa olevia mikrotehtäviä. Mikrotehtävien läpi vedetään suoritusviiva, joka kertoo, missä järjestyksessä mikrotehtävät suoritetaan. Jokin osa taitoa vaativan tehtävän suorituksessa voi olla sen verran hankalampi, että ei tiedetä, missä järjestyksessä tai kuinka moneen kertaan mikrotehtävät pitäisi suorittaa parhaan tuloksen saamiseksi. Tällöin vedetään laatikko (kuvassa "Learn order" eli "opi järjestys") niiden mikrotehtävien ympärille, joiden järjestyksen halutaan muuttuvan oppimisen aikana. Mikrotehtäviä voi olla alkuvaiheessa enemmänkin kuin tarvitaan. Niitä tai niiden määrää pystytään karsimaan myöhemmin oppimistilanteessa. Mikrotehtävien alasvetovalikoista (kuvassa kolmiot) voidaan valita suoritusten määrän suurin ja pienin arvo, jolloin "Move Arms" –mikrotehtävän suoritusmääriksi voidaan asettaa esimerkiksi 1-6. Lisäksi mikrotehtävien oheen voidaan liittää objekteja ja paikkoja (kuvassa laatikot "Object: Box" ja "Place: Start Point"), jotka voidaan antaa mikrotehtäville parametreina. Lopuksi tehtävän aloituksen ja lopetuksen merkiksi liitetään "Start"- ja "End"-laatikot mikrotehtäväjonon päihin.

5.3.2 Tehtävänanto mallisuorituksella

Mallisuoritusta näyttäessään opettaja voi käskyttää WorkPartneria joko puhekomennoilla tai teleoperointivaljailla. Puhekomennoilla voidaan sanoa WorkPartnerille "aloita tehtävän tallennus", jolloin kaikki seuraavat komennot tallennetaan osaksi opeteltavaa tehtävää. Puhekomennoilla voidaan sanoa myös esimerkiksi "Liiku eteenpäin! Pysähdy! Tuo kädet alas!" Puhutuista komennoista erotellaan käskyt toisistaan ja luodaan mikrotehtäviä. Jo puheen mukana voi antaa paljon lisäinformaatiota, kuten "liikuttaessa GA on päällä", mutta lisäinformaation voi lisätä myöhemminkin (esim. graafisella käyttöliittymällä). Puhekomentojen

tunnistaminen ei ole toistaiseksi kuitenkaan helppoa. WorkPartnerin pitäisi ymmärtää monimutkaisia komentoja, jotta tämänlainen tehtävänanto olisi käytännöllinen. Lisäksi opettajan olisi muistettava komentojen muodot tarkkaan, jotta robotti ne ymmärtäisi.

Teleoperointivaljailla voidaan WorkPartnerin liikkeitä ohjata paljon helpommin kuin puhekomennoilla. Niillä voidaan esimerkiksi liikuttaa WorkPartnerin manipulaattoreita yksityiskohtaisen tarkasti. Laite ei kuitenkaan sovellu monimutkaisten komentojen välittämiseen. Käsiä heiluttamalla on vaikea kertoa robotille esimerkiksi, että "tässä osatehtävässä ei käytetä oppimista". Teleoperointivaljaat sopivat paremmin tehtävänantoon, jos ne yhdistetään puhekomentoihin. Tällöin laitteilla annettaisiin pikkutarkkoja komentoja ja puheella monimutkaisempia komentoja. Esimerkiksi teleoperointivaljailla voitaisiin käskyttää robottia liikkumaan tietyllä tavalla ja nappia painamalla voitaisiin signaloida robotille mikrotehtävien vaihtumisesta. Näin saataisiin mikrotehtäväksi tallennettua liikettä, joka voidaan graafisen käyttöliittymän avulla tarkentaa ja muokata.

5.4 Oppiminen tehtävässä

5.4.1 Oppimisen aloitus

Kun robotille on selvää, mitä sen pitää eri tehtävän vaiheissa tehdä, voidaan aloittaa taitoa vaativan tehtävän varsinainen opettaminen. Tässä vaiheessa WorkPartner tietää, minkälaisia osa- ja mikrotehtäviä sen pitää suorittaa, mutta se ei tarkkaan tiedä, mitä tehdä näissä tehtävissä. Toisin sanoen siltä puuttuu tehtävien tarvitsemat parametrit eli alkuarvot. Alkuarvoilla tarkoitetaan ensimmäisten suoritusyritysten tarvitsemia parametrien arvoja, kuten manipulaattorien koordinaattipisteitä tai mikrotehtävien suoritusjärjestyksiä. WorkPartner voi joko antaa itselleen satunnaisia alkuarvoja tai opettaja voi antaa arvot. Robotin antaessa alkuarvot itse, saattaa oppiminen selkeästi hidastua, koska satunnaiset parametrit tuottavat yleensä huonoja suoritusyrityksiä. Alkuarvot saatuaan WorkPartner alkaa opetella tehtävän suoritusta, joko käyttäjän avustuksella tai itsenäisesti. Kun robotti on oppinut tehtävän suorituksen, voidaan se jättää itsenäisesti työskentelemään.

5.4.2 Opettamisvaihe

Opettamisvaiheessa käyttäjä antaa WorkPartnerille taitoa vaativan tehtävän alkuarvot. Opettajana toimivan käyttäjän avustuksella robotti saa alusta alkaen melko hyviä parametreja, mikä nopeuttaa onnistuneiden suoritusten löytämistä. Opettaja voi antaa alkuarvot esimerkiksi kuvassa 9 esitetyllä graafisella käyttöliittymällä. Hänen on ennalta tiedettävä, minkälaiset alkuarvot ovat hyviä, jotta ne olisivat parempia kuin satunnaiset arvot.

Graafista käyttöliittymää parempi keino antaa alkuarvoja on teleoperointivaljailla ja puhekäyttöliittymän avulla. Teleoperointivaljailla voidaan ohjata tarkkaan alustan liikkeitä, jolloin alkuarvot saadaan tallennettua kätevästi. Esimerkiksi "laatikon nosto" –tehtävässä voidaan antaa hyvät alkuarvot robotin manipulaattorien liikkeille näyttämällä mallia valjailla käsien liikeradoista laatikkoa nostettaessa.

"Laatikon nosto" –osatehtävässä teleoperointivaljailla on mahdollista saada kätevästi tallennettua tilasiirtymiä alkuarvoiksi. Käyttäjä voisi liikkua työympäristössä ja "suorittaa" liikkeillään tehtävää itse. WorkPartner olisi käyttäjän lähellä ja matkisi hänen tekemisiään. Käyttäjälle riittäisi kaksi tai kolme kytkintä tilasiirtymien tallentamiseksi mikrotehtävien välillä. Mikrotehtävästä toiseen voitaisiin siirtyä napin painalluksella, jolloin kuulokkeista tulisi ilmoitus seuraavasta mikrotehtävästä. Mikrotehtävän parametrien tallennus onnistuisi toisella napilla. Mahdollisesti kolmas nappi tarvittaisiin valintojen hyväksymiseen. Ollessaan laatikon edessä käyttäjä vaihtaisi mikrotehtäväksi käsien liikuttamisen ja tallentaisi napin painaluksella pari hyvää käsien asentoa. Seuraavaksi hän vaihtaisi mikrotehtäväksi vartalon kallistuksen ja teleoperointivaljailla kallistaisi WorkPartnerin kulmaa sopivaksi. Mikrotehtävä vaihdettaisiin sitten laatikon paikan mittaukseen ja nappia painamalla kerrottaisiin robotille, että nyt mitataan laatikon paikka. Seuraavaksi siirryttäisiin taas käsien liikuttamiseen ja annetaisiin loput käsien paikkapisteet WorkPartnerille, minkä jälkeen sen pitäisi pystyä kaappaamaan laatikko. Kun nämä mikrotehtävät suoritetaan peräkkäin, saadaan sulavaa liikettä, jossa käsiä liikutetaan ensin levälleen ja kohti laatikkoa, samalla vartaloa kallistaen eteenpäin, sitten laatikon paikka mitataan ja kädet jatkavat liikettään ottaen kiinni laatikosta. Lopuksi tallennetaan suoritus tilasiirtymien jonoksi ja voidaan siirtyä seuraavan osatehtävän opettamiseen.

On tärkeää muistaa, että jokaiselle populaatiolle on oltava omat alkuarvonsa. Opettamisvaiheessa pitää siis antaa populaation koon verran alkuarvoja jokaiselle mikrotehtävälle. Populaation koko kertoo esimerkiksi, montako mallisuoritusta robotille näytetään kustakin tehtävästä. Lisäksi alkuarvot eivät saa olla keskenään samoja tai edes lähes samoja. Geneettinen algoritmi ei saa samankaltaisista lukuarvoista generoitua erillisiä populaatioita, mikä pitkittää käyttökelpoisten suoritusyritysten löytymisaikaa. Tästä seuraa että algoritmilla kestää hyvin kauan saada mitään järkeviä suoritusyrityksiä. Tämä ongelma voidaan ratkaista tekemällä ohjelmaan satunnaisgeneraattori, jolla luodaan muutamasta mallisuorituksesta niistä poikkeavia lisäsuorituksia.

5.4.3 Oppimisvaihe

Kun alkuarvot on annettu mikrotehtäville, WorkPartner voi ruveta kehittämään eli oppimaan taitoa vaativan tehtävän suoritusta. Tässä vaiheessa on helppo huomata tehtävässä olevat puutteet ja korjata ne palaamalla takaisin tehtävänantovaiheeseen. Tehtävän suoritusyritysten taso on aluksi hyvin heikkoa, koska WorkPartner ei ole ehtinyt oppia vielä mitään. WorkPartnerin annetaan yrittää suorittaa tehtävää, jolloin osatehtävien suorituksille annetaan fitness-arvot. Fitness-arvoja hyödyntäen luodaan geneettisellä algoritmilla uusi populaatio suoritusyrityksiä, minkä jälkeen WorkPartner yrittää suorittaa tehtävää uudelleen. Fitness-arvon antaa joko käyttäjä tai robotti itse. Jotkut osatehtävät ovat sen verran suoraviivaisia, että niille on helppo keksiä fitness-arvojen kriteerit (esim. laatikon etsinnän fitness-arvo tulee helposti mitattavissa olevasta suorituksen nopeudesta). Toiset osatehtävät ovat niin monimutkaisia, että niille on hankala keksiä selkeitä kriteereitä, jolloin fitness-arvon antaa käyttäjä (esim. laatikon nostossa). Kun tehtävän suorituksen taso on noussut tarpeeksi korkeaksi ja käyttäjä on tyytyväinen suoritukseen, siirrytään tehtävän suoritusvaiheeseen.

5.4.4 Suoritusvaihe

Suoritusvaiheessa WorkPartner yrittää suorittaa tehtävää itsenäisesti. Käyttäjä ei enää puutu robotin työskentelyyn, korkeintaan valvoo, ettei mitään yllättävää tapahdu. Suoritusvaiheessakin robotissa voi tapahtua oppimista, jos käyttäjä niin haluaa. Oppiminen suoritusvaiheessa vaikuttaa vain niihin osatehtäviin, joille robotti voi itse antaa fitness-arvoja. Sen lisäksi oppimisen nopeus on hidasta ja populaatiota ei

vaihdeta uudempaan, jos edellisen populaation ratkaisut ovat uutta parempia. Tämä siitä syystä, että suoritusten yritykset voisivat äkillisesti muuttua huonompaan suuntaan, jos uusi muunnos saisi tehtävän epäonnistumaan ja siten palauttaisi oppimissyklin lähemmäs alkutilannetta. Oppiminen kannattaa estää suoritusvaiheessa, jos halutaan varmistua siitä että tehtävän suoritus ei muutu miksikään ja jos tiedetään, että ympäristössä ei tapahdu mitään tehtävän suoritukseen vaikuttavia muutoksia. Myös jos robottia ei voida tarkkailla kovin usein, kannattaa oppiminen estää. Jos jokin kuitenkin menee pieleen tehtävää suoritettaessa, WorkPartner pystyy ilmoittamaan vian käyttäjälle.

5.5 Monimutkaisten tehtävien opetus

Monimutkaisia taitoa vaativia tehtäviä voi olla hankala opettaa WorkPartnerille, koska niiden läpikäynti on hidasta. Lisäksi onnistuneita suorituksia esiintyy vain harvoin, koska yhden osatehtävän epäonnistuessa muutkin menevät pieleen. Vaikka jollakin osatehtävällä olisi hyvä suoritusyritys, ei sitä välttämättä nähtäisi muiden osatehtävien epäonnistumisien takia. Fitness-arvojen antaminen on tällöin hyvin hankalaa.

Monimutkaisia tehtäviä voidaan koostaa useammasta helpommasta tehtävästä. Aluksi opetetaan suoriutuminen yksinkertaisista taitoa vaativista tehtävistä. Sen jälkeen yksinkertaiset taitoa vaativat tehtävät yhdistetään (esim. graafisella käyttöliittymällä kuvassa 9) yhdeksi monimutkaisemmaksi taitoa vaativaksi tehtäväksi. Tällöin yksinkertaiset tehtävät ovat monimutkaisen tehtävän osatehtäviä. Monimutkaisen tehtävän harjoittamista pitää todennäköisesti jatkaa, vaikka osatehtävien suoritus onkin jo kunnossa, koska osatehtävät eivät välttämättä sovi hyvin yhteen (toinen osatehtävä ei osaa jatkaa siitä mihin toinen jäi).

5.6 Tehtävän suoritukseen vaikuttavia muita tekijöitä

5.6.1 Paikannus tehtävässä

Koska työssä tutkitaan, miten WorkPartner voisi toimia mahdollisimman samalla tavalla kuin eläin oppiessaan, ei robotin ole tarpeen tietää tarkkaa sijaintiaan koordinaatistossa. Eiväthän eläimetkään tiedä muuta kuin ympäristön paikkojen suhteellisia arvioituja sijainteja toisiinsa. Näin pyritään toimimaan myös WorkPartnerinkin tapauksessa.

Tässä työssä on pyritty vähentämään tarkan tiedon käyttöä tehtäviä suoritettaessa. Tarkkaa tietoa käytetään ainoastaan tarkistettaessa, onko suoritus onnistunut vai ei. Jos tehtävä pystytään suorittamaan osittain ilman paikannusta, voidaan tehtävä suorittaa silloin paremmin paikannusta käyttäen.

5.6.2 Kuvankäsittely

Tässä työssä keskitytään geneettisen algoritmin toteuttamiseen. Tästä syystä kuvankäsittelypuoli jätetään vähemmälle ja ohjelman kuvankäsittelyn pohjana käytetään Jouni Sievilän (2003) tekemän tunnistus- ja etsintäohjelman koodia hieman muokattuna.

6 Tilannehahmotelma työtehtävästä ja sen oppimisesta

Alla esitellään mahdollinen tilannehahmotelma, jossa käyttäjä opastaa WorkPartneria suorittamaan taitoa vaativan tehtävän. Tehtävänä on luvussa 5 esitetty tehtävä, jossa etsitään punainen laatikko, mennään sen luo, nostetaan se ja tuodaan takaisin. Tässä hahmotelmassa oletetaan, että WorkPartner ei osaa vielä suorittaa tätä tehtävää, eikä ennakolta tunne toimintaympäristöään.

6.1 Alkutilanne

WorkPartner tuodaan suureen tehdashalliin tekemään "laatikon nouto" –tehtävää. Tarkoituksena on kouluttaa WorkPartneria tulevaa vaikeampaa tehtävää varten, jossa sen pitää osata liikkua tehdashallissa vapaasti ja etsiä tiettyjä laatikoita. Jotta tässä vaikeammassa tehtävässä onnistuttaisiin, on ensin opittava suorittamaan "laatikon nouto" –tehtävä.

WorkPartnerilla on valmiiksi tietämystä perusoperaatioista, kuten liiku, etsi, nosta, laske ja seuraa, joita se osaa käyttää ilman suurempaa opetusta. WorkPartnerille luodaan uusi työtehtävä (esim. "nouda laatikko"), jossa määritetään, minkälaisiin osa- ja mikrotehtäviin tehtävä jakautuu. Tämä määritys tehdään graafisella käyttöliittymällä, jossa samalla asetetaan oppimiseen liittyviä parametreja, kuten populaation koko ja risteytyksien ja mutaatioiden todennäköisyydet.

Ennen kuin tehtävää voidaan alkaa suorittaa, robotille asetetaan sääntöjä ja rajoitteita joita sen kuuluu noudattaa. Nämä voisivat esimerkiksi olla ympäristön rajaaminen eli alue, minkä sisällä robotti saa liikkua tai tiettyjen laitteiden käyttökielto, jos alueella on herkkää tekniikkaa. Tässä tapauksessa robotille asetetaan käsien nostorajoitus (käsiä saa nostaa korkeintaan 1,9 metrin korkeuteen), koska hallissa on joidenkin käytävien välissä noin kahden metrin korkeudessa hyllykköjen välillä kulkevia sähkökaapeleita.

6.2 Työtehtävän oppiminen

Kun WorkPartner on tuotu tehtävän aloitusalueelle, sille aletaan opettaa tehtävää. Aluksi robotille näytetään tehtävässä käsiteltävää kappaletta eli punaista laatikkoa, jonka tiedot robotti tallentaa objektitietokantaansa myöhempää tunnistusta varten.

Robottia käsketään myös tallentamaan tämänhetkinen sijaintinsa kotipaikaksi. Tähän paikkaan palataan, kun laatikko on saatu haltuun.

Oppimisen aikana WorkPartner luo käyttäjän näyttämästä esimerkistä tai opastuksesta populaation ratkaisuja kokonaistehtävälle muokkaamalla mikrotehtäviä sellaisiksi, että ne vastaavat kokonaistehtävän suoritusta. Käyttäjä antaa robotille graafisella käyttäliittymällä joukon pään kääntöön liittyviä alkuarvoja. Seuraavaksi käyttäjä pukee päälleen teleoperointivaljaat ja näyttää WorkPartnerille kaksi laatikon nostotapaa. Käyttäjä tallentaa nämä suoritukset ja generoi käyttöliittymällä kahdesta mallisuorituksesta lisää erilaisia suoritusyrityksiä, kunnes populaatiossa on riittävästi alkuarvoja. Käyttäjä ei osaa sanoa miten tilasiirtymät tulisi valita, joten hän luo ne satunnaisesti, toteaa että ne eivät ole täysin mahdottomia ja tallentaa ne. Tämän jälkeen robotti yrittää suorittaa "laatikon nosto" –tehtävää itsenäisesti. Tehtävän suoritusyritysten taso on aluksi melko heikko, mutta siinä ei näytä olevan selkeitä puutteita, joten käyttäjällä ei ole tarvetta muokata WorkPartnerin suoritusyritysten lähtötilannetta miksikään.

Robotti kysyy käyttäjältään jokaisen tehtävän suoritusyrityksen jälkeen palautetta suorituksen tasosta. Käyttäjä antaa fitness-arvon robotin suoritukselle ja WorkPartner siirtyy seuraavan ratkaisun yrittämiseen. Kun kaikki populaation ratkaisuyritykset on käyty läpi, syöttää robotti ratkaisut geneettiseen algoritmiin, joka tekee fitness-arvojen avulla ratkaisuista uuden populaation ratkaisuyrityksiä. Uudet ratkaisuyritykset suoritetaan, kunnes nekin on käyty läpi ja taas geneettinen algoritmi tekee uuden populaation vanhan pohjalta. Tätä silmukkaa jatketaan, kunnes ratkaisuyritykset poikkeuksetta onnistuvat, jolloin nämä ratkaisut tallennetaan muistiin ja otetaan populaatiosta paras ratkaisu kuvamaan taitoa vaativan tehtävän suoritusta.

6.3 Työtehtävän suoritus

Kun tehtävää aletaan suorittaa, otetaan aluksi oppimismoodi pois päältä, jolloin robotti ei muuta enää ratkaisujaan. Koska robotti tulee työskentelemään itsenäisesti ympäristössä, jossa sitä ei jatkuvasti voida tarkkailla, on käyttäjän varmistuttava, että robotti toimii moitteettomasti tarkkailun alaisena. Robotti suoriutuu hyvin "laatikon nosto" –tehtävästä, joten käyttäjä kokeilee miten WorkPartner suoriutuu, kun siinä on oppimismoodi päällä. Koska WorkPartner on jo oppinut suoriutumaan tehtävästä hyvin, käyttäjä muuttaa oppimisen nopeutta hitaammaksi. Tällöin populaatioon

tehdään vain hyvin pieniä muutoksia ja populaatiota ei vaihdeta uudempaan, jos edellisen populaation ratkaisut ovat parempia kuin uuden. Tällä saadaan aikaan pientä hienosäätöä, jotta robotti sopeutuisi mahdollisimman hyvin ympäristöönsä.

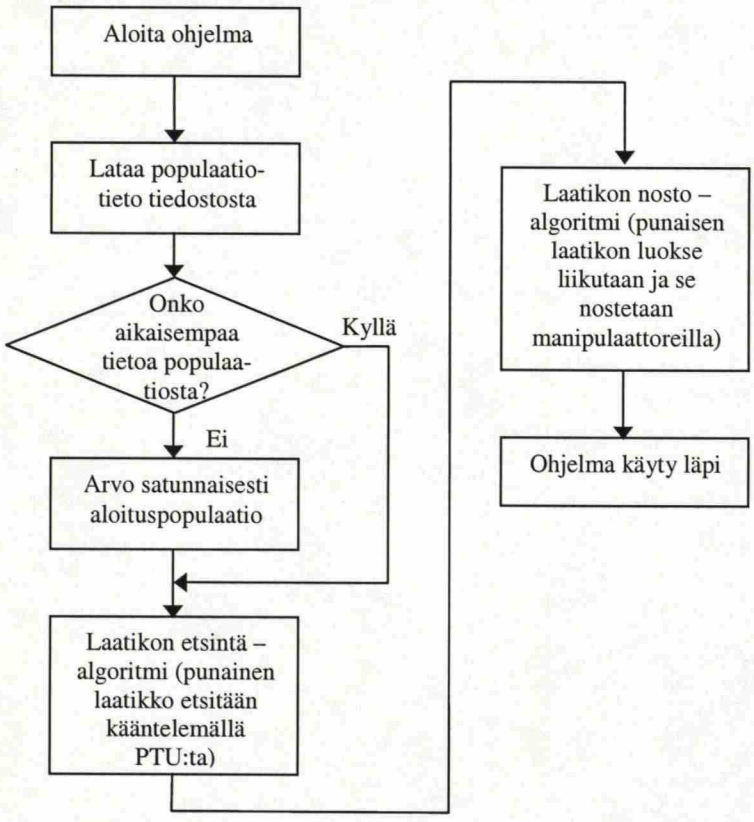
Oppimisen jälkeen käyttäjä on saanut hiottua tehtävänsuorituksen todella hyväksi. Käyttäjä näyttää tehtävänsuorituksen tason työtovereilleen arviointia varten. Käyttäjä antaa robotille käskyn aloittaa työtehtävä. Käskyn saatuaan WorkPartner etsii päätään kääntelemällä ympäristöstään punaisen laatikon. Robotti mittaa laatikon paikan ja etäisyyden laser-etäisyysmittarilla. Robotti ajaa laatikon luo suorinta mahdollista reittiä, esteet kiertäen ja pyrkii pääsemään "riittävän" lähelle laatikkoa. Laatikon luona robotti mittaa paikkansa laatikon suhteen uudelleen, kallistaa yläruumistaan eteenpäin riippuen laatikon sijainnista ja rupeaa viemään käsiään levällään kohti laatikkoa. Robotin kädet tekevät kaappausliikkeen laatikkoa kohden ja robotti saa laatikon käsiensä väliin. Tämän jälkeen laatikkoa puristetaan sopivasti, jotta se pysyy käsissä ja laatikko nostetaan ilmaan. Seuraavaksi robotti palaa laatikko käsissään paikkaan, mistä se lähti ja sijoittaa laatikon maahan. WorkPartner ilmoittaa, että "laatikon nosto" –tehtävä on suoritettu. Työtoverit antavat hyväksyviä hymähdyksiä käyttäjälle ja WorkPartnerille. Seuraavan työtehtävän opetus voi alkaa.

6.4 Muita sovelluksia

Edellä esitetystä tilannehahmotelmasta saadaan soveltamalla aikaan monenlaisia muitakin taitoa vaativia tehtäviä, jotka käyttävät hyvin pitkälle aivan samoja osa- ja mikrotehtäviä kuin kohdissa 6.1 – 6.3 on esitetty. Esimerkiksi tehtävät, joissa tavarat pitää järjestää tietyllä tavalla, toteutetaan lisäämällä tähän tehtävään vain useita etsittäviä ja nostettavia kappaleita ja paikkoja joihin kappaleet viedään. Muita esimerkkejä ovat erilaiset siivoustehtävät, tiettyjen tavaroiden keräilytehtävät tai oikeastaan, mitkä tahansa "mennään johonkin kohteeseen, otetaan jotain kantoon ja mennään toiseen kohteeseen" –tyyppiset tehtävät.

7 Toteutus

Diplomityössä tehtiin ohjelmaa, jolla voidaan testata yhden taitoa vaativan tehtävän suorituksia. Toteutettu taitoa vaativa tehtävä on luvussa 5 esitetty laatikon etsintä- ja nostotehtävä. Ohjelmaa tehtiin C++ Builder –ohjelmalla. Ohjelman käyttöliittymästä on esitys liitteessä 1. Kuvassa 10 on esitetty ohjelman toiminnan rakenne.



Kuva 10. Diplomityössä toteutetun ohjelman rakenne.

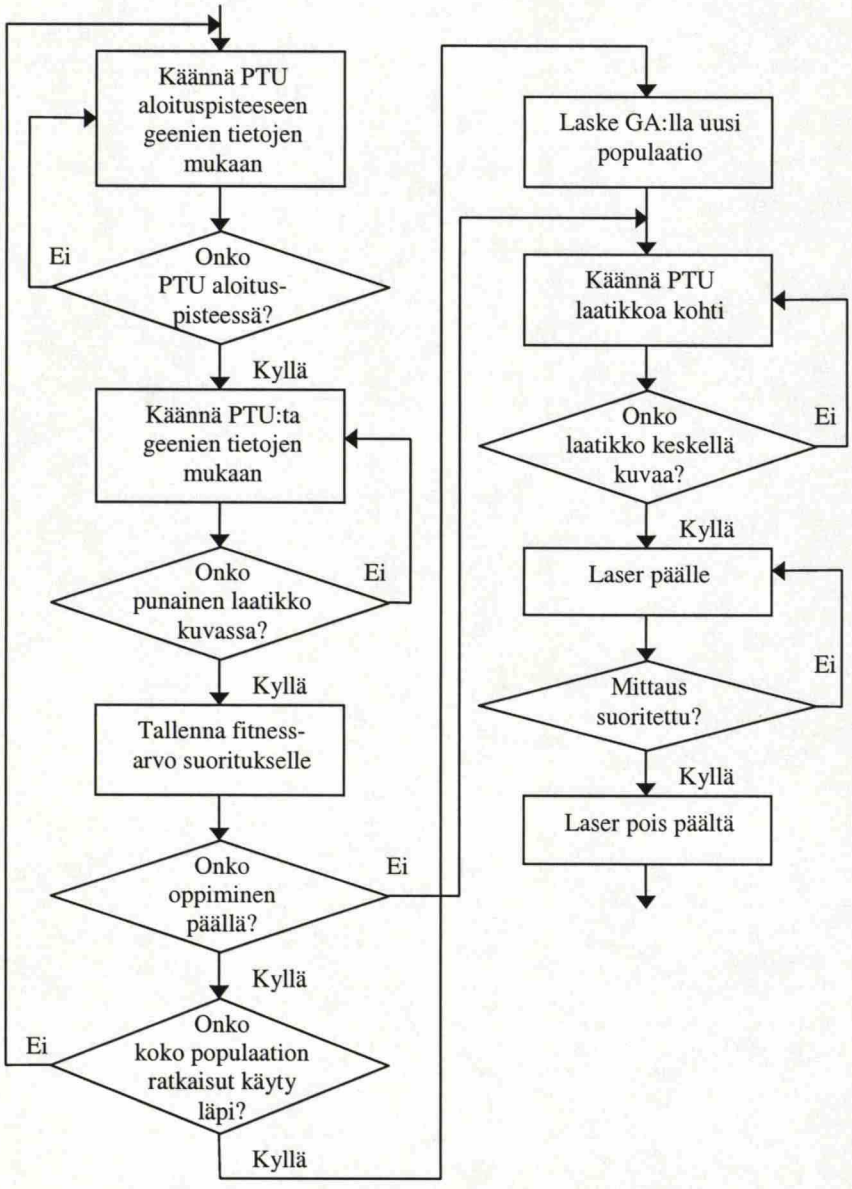
Kuten kuvassa 10 nähdään, ohjelma jakautuu selkeästi kahteen osaan, "laatikon etsintään" –algoritmiin ja "laatikon nostoon" –algoritmiin. Kummassakin osassa oppiminen on toteutettu hieman eri tavalla kuin toisessa. Algoritmit esitetään tarkemmin kohdissa 7.1 ja 7.2.

7.1 Laatikon etsintä –algoritmin toteutus

7.1.1 Algoritmin kuvaus

"Laatikon etsintä" –algoritmissa WorkPartner etsii punaista laatikkoa. Etsintä tapahtuu WorkPartnerin päätä eli PTU-yksikköä kääntelemällä puolelta toiselle sekä

ylös ja alaspäin. Kun laatikko on löydetty, sen etäisyys ja paikka mitataan laser-etäisyysmittarilla. Tehtävän fitness-kriteerinä on nopeus, joten nopeat tehtävän suoritukset pärjäävät geneettisen algoritmin valinnassa. Tehtävänsuorituksen parantuessa laatikon etsintä nopeutuu siten, että algoritmi oppii katsomaan ensin ne paikat, missä laatikko todennäköisimmin on. Esimerkiksi lattiapinnat etsitään ennemmin kuin katsotaan katon rajasta. Kuvassa 11 on esitetty "laatikon etsintä" –algoritmi kaaviona.



Kuva 11. "Laatikon etsintä" –algoritmin rakenne.

"Laatikon etsintä" –algoritmissa populaatiossa olevat parametrit kertovat algoritmille, kuinka päätä eli PTU:ta käännellään. Algoritmissa PTU:n kääntelylle on annettu säännöt, joita sen pitää noudattaa. Algoritmin oppimisosuus on kuvattu alla.

"Laatikon etsintä" –algoritmin oppimisosuus:

1. WorkPartner kääntää päänsä aloituspisteeseen (begin point).
2. WorkPartner liikuttaa päätään joko vaaka- tai pystysuuntaan skannaten yhden linjan reunasta reunaan. Parametri *order* kertoo bittijonomuodossa, missä järjestyksessä linjat käydään läpi.
3. Ohjelma tallentaa aina, mitkä linjat on käyty läpi ja kertoo WorkPartnerille "säännöt", miten ympäristöä skannataan ("ensin oikealle sitten vasemmalle").
4. Kun punainen laatikko löytyy tai yksi skannaus on tehty, tallennetaan skannauksen suoritus ja fitness-arvo muistiin ja siirrytään seuraavaan populaation ratkaisuun.
5. Palataan alkuun (kohtaan 1) kunnes koko populaation ratkaisut on käyty läpi.
6. Seuraavaksi ratkaisuista valitaan Ruletti-valinnalla keskenään risteytettävät ratkaisut ja luodaan uusi populaatio ratkaisuja. Algoritmi aloitetaan alusta (palataan kohtaan 1).

Päädyin tähän "laatikon etsintä" –ratkaisuun, koska se on tarpeeksi yksinkertainen, jotta ihminen pystyy intuitiivisesti näkemään, lähestyykö robotti hyvää ratkaisua. Samalla se on vielä tarpeeksi monimutkainen, jotta siinä voi syntyä suuri määrä erilaisia ratkaisuja. Lisäksi se pystytään toteuttamaan helposti GA:ta käyttäen. Jos oltaisiin käytetty sellaista etsintämekanismia, jossa robotti olisi saanut kääntää päätään mihin paikkaan tahansa, olisi ratkaisuissa katsottu samoja alueita useaan otteeseen PTU:n kääntyillessä edestakaisin. Lisäksi parametreistä olisi tullut eri pituisia, jolloin niiden käsitteleminen geneettisessä algoritmissa (esim. risteytyksessä) ei olisi onnistunut.

7.1.2 Parametrit

Ohjelmalla tehdään aluksi satunnainen populaatio ratkaisuja. Ratkaisuille arvotaan skannausjärjestys, vaaka- ja pystysuuntaisten linjojen määrät ja skannauksen

aloituspiste. Oppimisessa käytetyt parametrit on kuvattu alla. Parametrien koodauksesta kerrotaan tarkemmin kohdassa 7.4 geneettisen koodauksen yhteydessä.

Oppimisen parametrit:

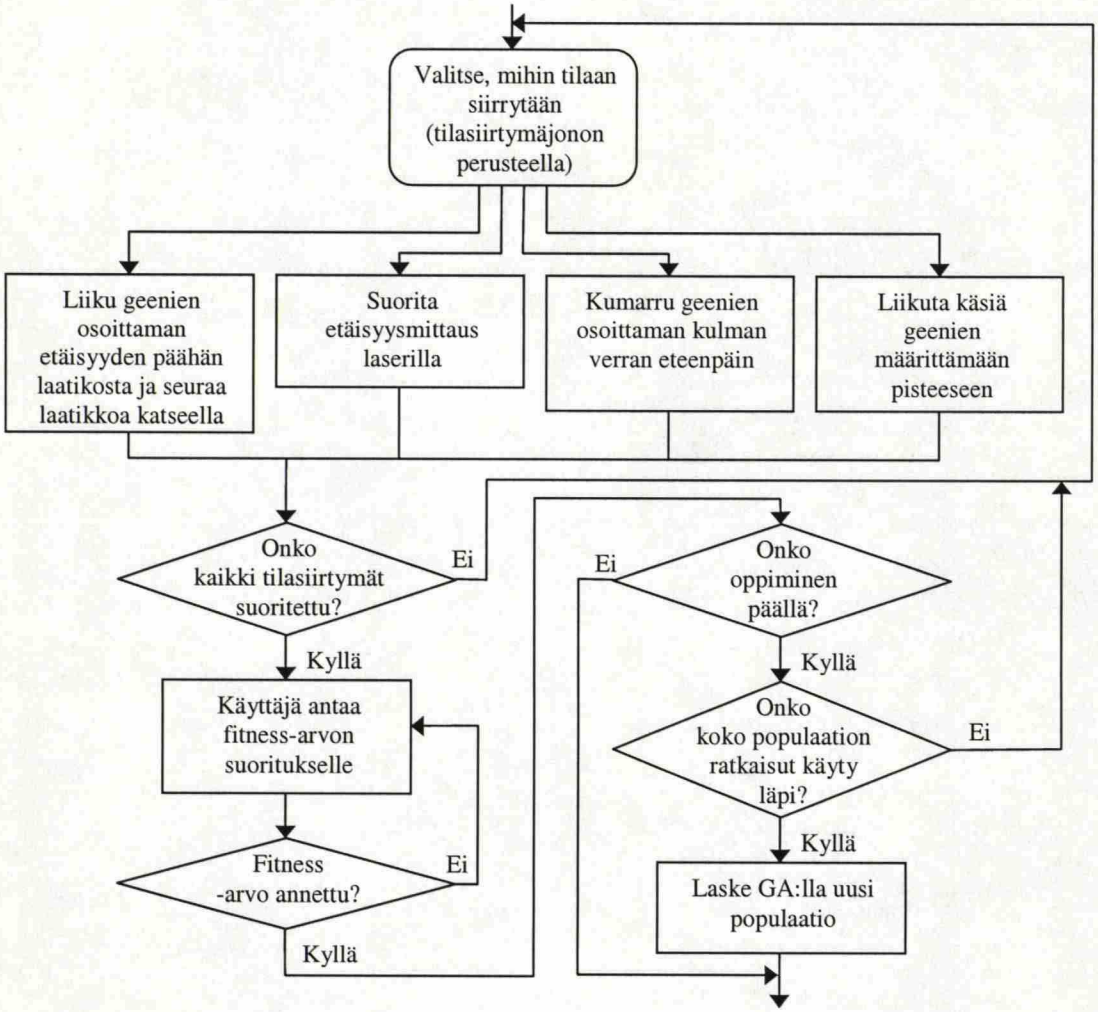
- Katselulinjojen läpikäyntijärjestyksen kertova parametri (nimeltä *order*) on bittijono (20 bittiä).
- Vaaka- ja pystysuuntaisten linjojen määristä kertovat parametrit (nimeltään *horizontal* ja *vertical*) ovat kokonaislukuja (*horizontal* on maksimissaan 5 ja *vertical* on maksimissaan 9, yhteensä vähintään 9).
- Katselun aloituspisteen koordinaatit (*beginX* ja *beginY*) ovat kokonaislukuja (*beginX* on välillä -4 ja +4, *beginY* on välillä -2 ja +2).

7.2 Laatikon nosto –algoritmin toteutus

7.2.1 Algoritmin kuvaus

Kun laatikko on löytynyt, WorkPartner pyrkii pääsemään tartuntaetäisyydelle. WorkPartner yrittää nostaa laatikon viemällä kätensä sivukautta laatikon ympärille, sulkemalla kätensä laatikon ympärille ja nostamalla käsiä ylöspäin. Koska tämän osatehtävän suorituksessa ei ole selvää, missä järjestyksessä osa mikrotehtävistä suoritetaan, on oppimisessa myös muokattava tilasiirtymäjonoja. Tilasiirtymäjonot kertovat, missä järjestyksessä yllä mainitut mikrotehtävät suoritetaan. Tilasiirtymäjonoista kerrotaan tarkemmin tilakoneen ja geneettisen koodauksen yhteydessä kohdissa 7.3 ja 7.4. Kuvassa 12 on esitetty "laatikon nosto" –algoritmi kaaviona.

"Laatikon nosto" –tehtävässä fitness-arvoja ei voida antaa yhtä helposti kuin "laatikon etsintä" –tehtävässä. Koska ei ole yksiselitteisen selvää mitkä olisivat hyvät kriteerit fitness-arvoille monimutkaisessa tehtävässä, jätetään suorituksen tason arviointi käyttäjälle. Yksinkertaisia ehtoja (onnistui / ei onnistunut) laatikon noston onnistumisen tarkistamiseksi pystytään tekemään, mutta onnistumisen hyvyyttä on hankala määritellä yksiselitteisesti. Laatikon pysymistä käsien välissä voidaan mitata manipulaattorien virranmuutoksesta (virran kulutus on suurempi, kun laatikkoa joudutaan puristamaan).



Kuva 12. "Laatikon nosto" –algoritmin rakenne.

"Laatikon nosto" –algoritmissa populaatiossa olevat parametrit kertovat algoritmille, kuinka alustaa liikutellaan, jotta laatikosta saataisiin ote. Algoritmissa manipulaattorien liikuttelua on rajoitettu, että se ei vahingoittaisi itseään törmäilemällä käsillään muihin ruumiinsa osiin. Alustan liikkuminen ympäristössä on rajoitettu pyörillä liikkumiseen, joten käveleminen vaikeassa maastossa ei kehitystyön tässä vaiheessa vielä onnistu. Algoritmin oppimisosuus on kuvattu alla.

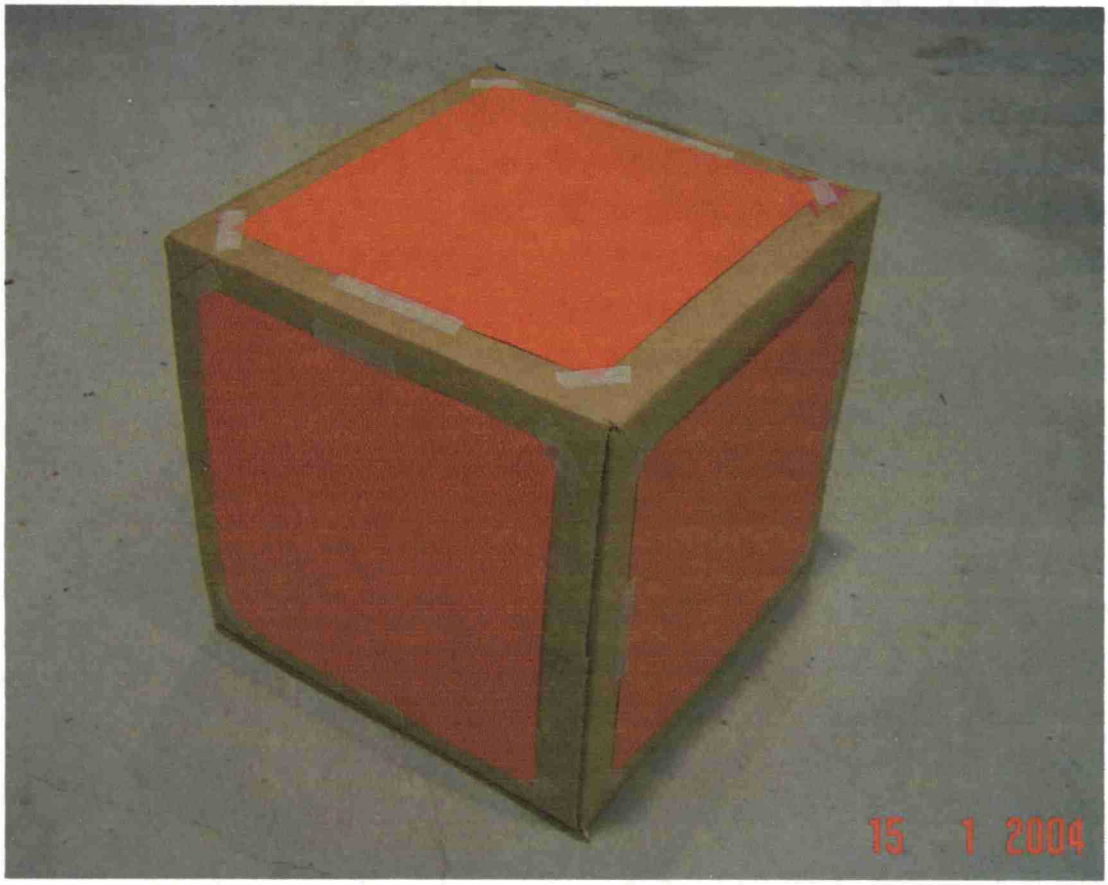
"Laatikon nosto" –algoritmi oppimisosuus:

1. Tilasiirtymäjono kertoo, mihin seuraavista allaolevista mikrotehtävistä (merkitty pallolla) siirrytään. Kun kaikki tilasiirtymäjonon osoittamat tilat on käyty läpi, siirrytään kohtaan 2.
    - WorkPartner lähestyy tietylle populaation osoittamalle etäisyydelle laatikosta.
    - WorkPartner mittaa laseretäisyysmittarilla laatikon paikan ja etäisyyden.
    - WorkPartner kallistaa vartaloaan populaation osoittaman kulman verran eteenpäin päästääkseen lähemmäs laatikkoa.
    - WorkPartner vie kätensä tiettyyn läpikäyntipisteeseen tai tartuntapisteeseen (tartuntapiste sijaitsee laatikon sisällä). Ensin suoritetaan läpikäyntipisteeseen viennit yksitellen (muita mikrotehtäviä voidaan suorittaa välillä) ja viimeiseksi suoritetaan käsien vienti tartuntapisteen.
2. Kun WorkPartner "tuntee" tai tietää ottaneensa laatikon kiinni, se nostaa laatikon ilmaan.
3. Suorituksen jälkeen tallennetaan ratkaisun fitness-arvo muistiin sen perusteella, kuinka hyvin "nosto" onnistui (putosiko laatikko vai ei ja kuinka tehokkaasti tehtävä suoritettiin). Fitness-arvon antamiseen tarvitaan ihmisen antamaa palautetta.
4. Palataan alkuun (1:een) ja suoritetaan seuraavan populaation ratkaisu, kunnes koko populaation on käyty läpi. WorkPartner peruuttaa ensin alkutilaansa.
5. Seuraavaksi suoritysyrityksistä valitaan rulettipyörävalintamenetelmällä keskenään lisääntyvät ratkaisut ja luodaan uusi populaatio ratkaisuja. Algoritmi aloitetaan alusta (palataan kohtaan 1).

Algoritmissa pyritään siihen, että viiveitä olisi mahdollisimman vähän mikrotehtävien välillä. Algoritmin kohdan 1 mikrotehtävät (merkitty palloilla) pyritään suorittamaan osittain päällekkäin siten, että esimerkiksi WorkPartnerin lähestyessä laatikkoa, se alkaa samalla jo liikuttaa käsiänsä kohti läpikäyntipisteitä ja asettamaan vartalonsa kulmaa sopivaksi. Kun WorkPartner on pysähtynyt laatikon eteen, se jatkaa

välittömästi käsiensä liikettä kohti tartuntapistettä ottaakseen kiinni laatikosta ja jatkaa edelleen nostamalla laatikon ilmaan.

Laatikkona tehtävässä käytettiin noin 30 cm halkaisijaltaan olevaa kuution muotoista pahvilaatikkoa. Laatikon sivuihin oli kiinnitetty punaiset paperit tunnistusta varten. Laatikko on tarpeeksi jykevä ja joustava kestämään WorkPartnerin manipulaattorien puristusvoiman. Laatikko kestää hyvin kovakouraistakin käsittelyä ja siksi sopii hyvin taitoa vaativan tehtävän testilaatikoksi. Kuvassa 13 näytetään kyseinen laatikko.



Kuva 13. Työssä käytetty "punainen" testilaatikko.

7.2.2 Parametrit

Ohjelmalla tehdään edellisen osasuorituksen tapaan satunnainen populaatio ratkaisuja tai sitten parametrit voidaan antaa manuaalisesti kirjaamalla ne populaation tallennustiedostoon. Ratkaisuille pitää antaa etäisyysarvo, kumarruskulma, läpikäyntipisteet, tartuntapisteet sekä tilasiirtymäjonot. Oppimisessa käytetyt parametrit on kuvattu alla.

Oppimisen parametrit:

- Liikkumisessa laatikon ja robotin väliin jätettävä etäisyydestä kertova parametri (nimeltä *box_distance*) on kokonaisluku (millimetrejä).
- WorkPartnerin vartalon kumarruskulman kertova parametri (nimeltä *body_tilt*) on kokonaisluku (asteita).
- WorkPartnerin käsien läpikäyntipisteistä kertovat parametrit (nimeltä app_point1_right, app_point1_left, app_point2_right, jne.) ovat kokonaislukuja (koordinaattipiste).
- WorkPartnerin käsien tartuntapisteestä kertova parametri (nimeltä grap_point) on kokonaisluku (koordinaattipiste).
- WorkPartnerin mikrotehtävien tilojen siirtymisistä kertova parametri (nimeltä *statepopulation*) on kokonaislukutaulukko.

7.3 Tilakone

7.3.1 Tilakoneen rakenne

Tässä työssä taitoa vaativa tehtävä on ohjelmoitu yhdistelmäksi erilaisia tiloja ja niiden välisiä siirtymiä. Yhden taitoa vaativan tehtävän suoritus voidaan mieltää tilakoneeksi, jossa yksi tila vastaa yksittäistä toimintoa. Tilakone on kuitenkin staattinen rakenne, jonka siirtymät tilasta toiseen eivät muutu. WorkPartnerin on oppiessaan taitoa vaativia tehtäviä kyettävä muokkaamaan tilakoneessa tiettyjen tilojen tilasiirtymiä. Tämä tarkoittaa sitä, että robotti oppimisensa aikana saattaa muuttaa näiden tilojen suoritusjärjestystä ja jättää joissakin tiloissa käymisen kokonaan väliin. Näitä tilojen siirtymisiä nimitetään tilasiirtymäjonoiksi. Tilojen siirtymien muuttaminen koskee vain niitä tiloja, jotka ovat vuorovaikutuksessa jollakin tavalla ympäristön kanssa (liikkuminen, etäisyyden mittaus) ja joiden

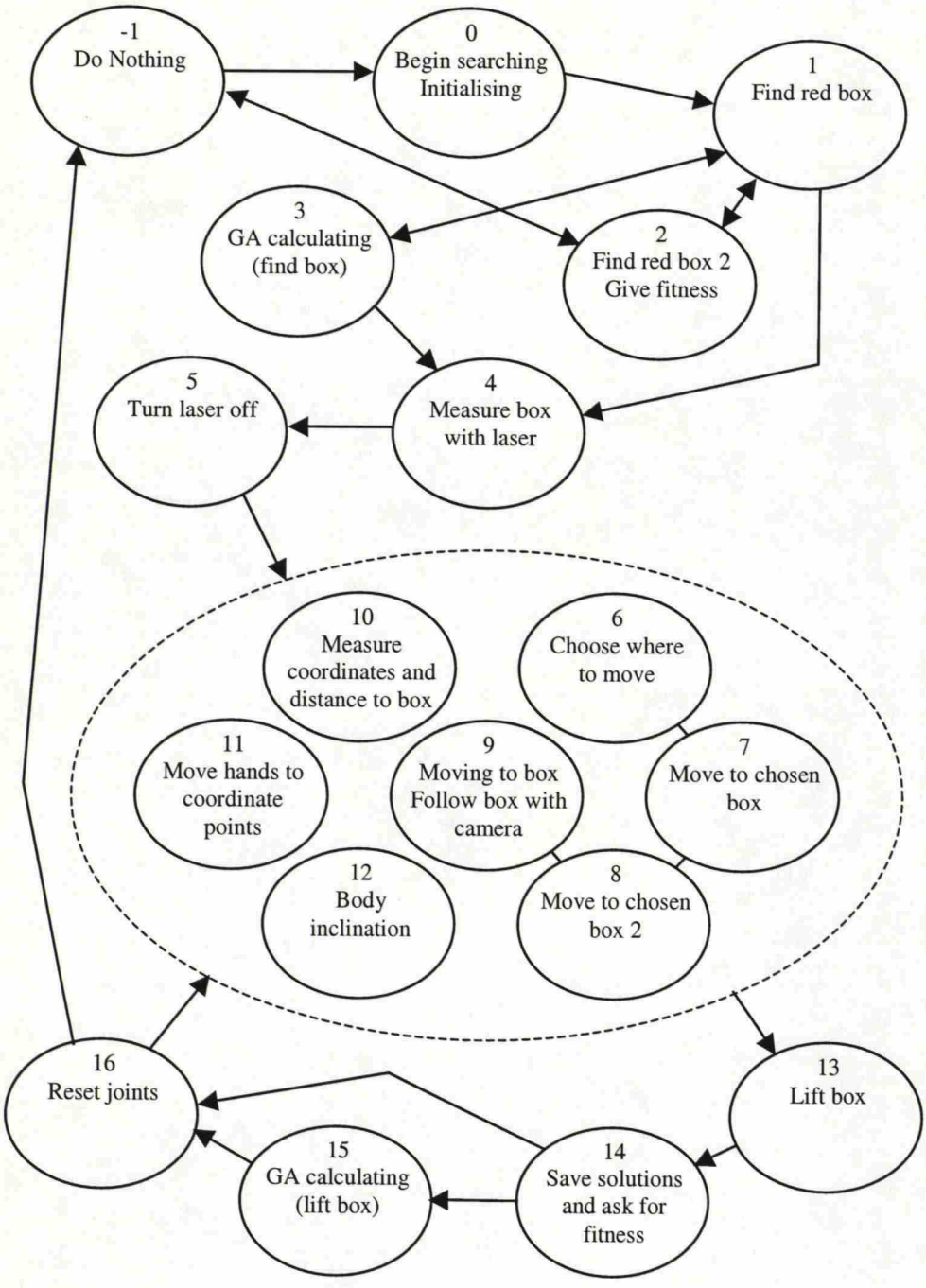
järjestyksen muuttaminen aiheuttaa muutoksia WorkPartnerin toimintaan. Näitä vuorovaikutteisia tiloja kutsutaan mikrotehtäviksi. Jotkut tilat vastaavat yhtä mikrotehtävää, toiset taas ovat vain pieni osa mikrotehtävää.

Tässä työssä tutkitussa taitoa vaativassa tehtävässä on neljä osatehtävää "etsi punainen laatikko, mene sen luokse, nosta se ylös ja vie se johonkin paikkaan", jotka taas jakautuvat erinäisiin mikrotehtäviin. Tämän työn ohjelmassa neljäs osatehtävä "vie laatikko johonkin paikkaan" on jätetty pois ja toinen osatehtävä "mene laatikon luokse" on yhdistetty kolmannen osatehtävän "nosta laatikko" kanssa. Syyt tähän muutokseen on selitetty kohdassa 5.1. Mikrotehtävien määrä ja järjestys voi muuttua vain "laatikon nosto" –tehtävässä. Näin voidaan testata "laatikon nosto" –tehtävässä, kuinka oppiminen onnistuu muuttuvalla tilakoneella ja "laatikon etsintä"-tehtävässä, kuinka oppiminen onnistuu normaalilla staattisella tilakoneella.

7.3.2 Tilakoneen tilat

Alla on esitetty kuvan 14 avulla ohjelman tilakoneen kaikki tilat. Yksi numeroitu ovaali kuvaa yhtä tilaa. Jokaisessa tilassa on englanninkielinen selostus tilan tapahtumista. Tilojen välillä on nuolia, jotka kertovat mahdollisista tilojen välisistä siirtymistä. Ohjelman käynnistys alkaa normaalisti tilasta nolla.

Kuvassa 14 näkyy katkoviivalla rajattu alue, jossa tilojen välillä ei ole nuolia. Tämä viittaa aikaisemmin mainittuun (alikohdassa 7.3.1) tilasiirtymien muokattavuuteen. Rajatun alueen mikrotehtävistä voi liikkua mihin tahansa muihin alueen mikrotehtäviin. Mikrotehtävät vastaavat usein yhtä tilaa, mutta kuvan 14 tilakoneessa tilat 6-9 kuuluvat kaikki samaan mikrotehtävään (kuvassa tilat on ketjutettu viivoilla yhteen). Kaikkia mikrotehtäviä ei tarvitse suorittaa ja joitakin voidaan suorittaa useamminkin kuin kerran. Kun tiloissa on käyty populaation geenien osoittamassa järjestyksessä, siirrytään rajatulta alueelta pois "Lift box" –tilaan (numero 13). Tilasiirtymien koodauksesta kerrotaan alikohdassa 7.4.2.



Kuva 14. Tehtävän opetusohjelman tilakoneen tilat ja tilasiirtymät.

7.4 Geneettinen koodaus

Jotta tehtävissä tapahtuisi oppimista, on mikrotehtävien parametreja muutettava geneettisellä algoritmilla. GA:ta varten parametrien on oltava tietyssä muodossa, jotta niille voitaisiin tehdä geneettisiä operaatioita (risteytys ja mutaatio). Tässä ohjelmassa toteutettu GA ymmärtää normaalisti vain vakiomittaisia bittijonoja. Toteutin

ohjelmaan lisäksi algoritmin, joka ymmärtää myös vakiomittaisia kokonaislukujonoja ja pystyy tekemään niille geneettisiä operaatioita.

7.4.1 Parametrien käsittely

Parametrit koodattiin sen mukaan, miten niiden haluttiin kehittyvän geneettisessä algoritmissa. Jos parametrilla on suuri joukko sallittuja arvoja, kuten koordinaattipisteillä, kannattaa parametri koodata bittijonoksi. Tällöin geneettiset operaatiot (risteytys ja mutaatio) voivat muuttaa parametrin arvoa koko sallitulla alueella. Jos taas parametrilla on vain tiettyjä sallittuja arvoja (esim. arvot eivät ole edes peräkkäisiä), kuten kokonaislukujonoilla, kannattaa parametri koodata (eli jättää) kokonaislukujonoksi. Tällöin geneettisillä operaatioilla muutettaessa parametrien arvoja, ne eivät lähde heittelehtimään sallitun alueen ulkopuolelle. Tämä johtuu siitä, että kokonaislukuja ei koodata bittimuotoon, jolloin niissä on aina samoja lukuja kuin vanhemmillaan (joista ne risteytettiin).

Ohjelmassa parametrit koodattiin geneettisiä operaatioita varten seuraavasti:

- Katselulinjojen läpikäyntijärjestyksen kertova parametri (nimeltä *order*) koodattiin bittijonoksi (20 bittiä) eli sen muotoa ei muutettu.
- Vaaka- ja pystysuuntaisten linjojen määristä kertovat parametrit (nimeltään *horizontal* ja *vertical*) koodattiin kokonaislukujonoksi.
- Katselun aloituspisteen koordinaatit (*beginX* ja *beginY*) koodattiin kokonaislukujonoksi.
- Liikkumisessa laatikon ja robotin väliin jätettävä etäisyydestä kertova parametri (nimeltä *box_distance*) koodattiin bittijonoksi (10 bittiä).
- WorkPartnerin vartalon kumarruskulman kertova parametri (nimeltä *body_tilt*) koodattiin bittijonoksi (10 bittiä).
- WorkPartnerin käsien läpikäyntipisteistä kertovat parametrit (nimeltä app_point1_right, app_point1_left, app_point2_right, jne.) koodattiin bittijonoksi (10 bittiä per parametri).
- WorkPartnerin käsien tartuntapisteestä kertova parametri (nimeltä grap_point) koodattiin bittijonoksi (10 bittiä per parametri).

- WorkPartnerin tilasiirtymistä kertova parametri (nimeltä statepopulation) koodattiin bittijonoiksi (5 bittiä per tila). Tilasiirtymäjonossa on maksimissaan 13 tilaa (tarvitaan siis 5*13 = 65 bittiä per jono).

7.4.2 Tilasiirtymien käsittely

Tilojen siirtymät tallennetaan tilasta toiseen siirtymisjärjestyksen kertovaksi numerojonoksi eli tilasiirtymäjonoksi. Samalla kun WorkPartner parantaa yksittäisen osatehtävän suoritusta geneettisen algoritmin avulla, parantaa se myös jokaisella sukupolven kierroksella tilasiirtymien jonoa paremmaksi. Alla on esitetty tilasiirtymäjonon ominaisuuksia ja kehittymisalgoritmi.

Ominaisuuksia:

- Tilasiirtymäjonossa on kaikkien yhdessä ratkaisussa käytävien tilojen suoritusjärjestys (esimerkki jonosta: 4536436660000).
- Jokainen sallittu tila (mikrotehtävä) voi esiintyä tilasiirtymäjonossa vain tietyn määrän verran. Esimerkiksi tila 6 ("liikuta käsiä" –tila) voi esiintyä 1-6 kertaa jonossa.
- Tilasiirtymäjono on vakiomittainen (13 lukua), jotta sitä olisi mahdollisimman helppo käsitellä kehittymisalgoritmilla.
- Koska jono on vakiomittainen ja tilojen määrä voi kuitenkin vaihdella, tyhjiä tiloja merkitään nollalla (kuten esimerkissä 4536436660000).
- Käyttäjä antaa ensimmäisen tilasiirtymäjonon robotille muiden parametrien antamisen yhteydessä.

Kehittymisalgoritmi:

1. Valitaan parhaimpien ratkaisujen tilasiirtymäjonot fitness-arvojen mukaan samaan tyyliin kuin geneettisellä algoritmilla.
2. Geneettisen algoritmin lisääntymisvalintojen mukaan tehdään kahden vanhemman tilasiirtymäjonosta uusi jono.
3. Käydään läpi kaikki vierekkäiset parit järjestyksessä. Vierekkäiset tilat vaihtavat paikkaa 0,3 todennäköisyydellä. Näin saadaan vaihtelua suoritusjärjestykseen.

4. Vierekkäisistä saman tilan tapahtumista toinen poistetaan 0,2 todennäköisyydellä (arvotaan, kumpi tiloista poistetaan). Näin karsitaan hiljalleen mahdollisia turhia tiloja pois.
5. Uusien suoritusten taso testataan ja aloitetaan algoritmi alusta.

7.5 Ohjelman kehittäminen

Tällä ohjelmalla on tutkittu yhtä taitoa vaativaa tehtävää (laatikon etsintä- ja nostotehtävää). Ohjelmaan on pyritty valitsemaan tehtävästä osia, joissa on mahdollisimman paljon erilaisia ominaisuuksia ja tapoja suorittaa niitä. Kun tämän tehtävän tutkiminen saadaan päätökseen, voidaan ohjelman pohjalta tehdä helposti muita taitoa vaativia tehtäviä. Ohjelmaa voitaisiin myös jatkokehittää oppimista hallinnoivaksi sovellukseksi, jolloin sitä voitaisiin käyttää uusien taitoa vaativien tehtävien opettelemiseen (samaan tapaan kuin kuvassa 9 esitetyssä graafisessa käyttöliittymässä).

8 Pohdintaa ja jatkotoimenpiteet

Oppiminen WorkPartner-robotilla on suuri haaste. Jotta oppiminen onnistuisi, pitää WorkPartnerin mekaniikan, elektroniikan ja ohjelmien suorituksen toimia ynteistyössä ilman suurempia ongelmia. Tämä on ensimmäinen kerta kun WorkPartnerilla on yritetty toteuttaa taitoa vaativien tehtävien oppimista. Oppimisen mahdollistavaa ohjelmaa tehtäessä tuli vastaan lukuisia vaikeita pulmia, joiden ratkaiseminen vaatii pohtimista. WorkPartner on kuitenkin kehittynyt huomattavasti muutaman viime vuoden aikana. Robotin mekaniikka ja elektroniikka on jo hyvässä kunnossa, joten suurin haaste lienee juuri ohjelmoinnin puolella. Painotusalueet aikaisemmissa tutkimuksissa WorkPartnerilla ovat lähinnä liittyneet kohteen tunnistamiseen, tilassa liikkumiseen ja navigointiin. Nyt ollaan siirtämässä tutkimuksia enemmän halutun kohteen etsintään ja löydetyn kohteen hallittuun käsittelyyn.

Taitoa vaativien tehtävien toteutuksia suunnitellessa tuli ohjelmoinnissa vastaan erilaisia ongelmia, joita on esitetty kohdassa 8.1. Näiden ratkaiseminen ei ole mitenkään mahdotonta, vaikkakin vaativat testaamista ja analysointia ongelmien ymmärtämiseksi.

8.1 Vastaan tulleita ongelmia

8.1.1 Tekniikkaan liittyviä ongelmia

WorkPartner saa ympäristöstään loppujen lopuksi kuitenkin hyvin vähän palautetta. Robotilla on muutama hyvin tärkeä aistinlaite (kamera, laser-etäisyysmittari ja laser-skanneri), joiden avulla se tekee lähes kaiken päättelyn ympäristöstään. WorkPartnerista puuttuu lähes kokonaan esimerkiksi eläimillä olevat hyvin tärkeät aistit kuten kuulo ja tunto. WorkPartnerille voidaan kyllä antaa puhekomentoja mikrofonin kautta, mutta se ei osaa kuunnella ympäristöään ja havaita asioita kuulon perusteella.

Eläimille hyvin tärkeä lähellä toimiva aisti on tunto. Sen avulla saadaan suunnaton määrä palautetta kaikesta mihin kosketaan. WorkPartner ei esimerkiksi huomaa, jos sen käsi törmää johonkin ja se voi vahingossa hajottaa itseään tai ympäristöään. Jos WorkPartneriin pystyttäisiin toteuttamaan kuulo- ja tuntoaistit, oppiminen helpottuisi

selkeästi paremman palautteen saamisen ansiosta. Myös fitness-kriteerejä voitaisiin parantaa, jolloin itseoppimisen toteuttaminen olisi helpompaa. "Tuntoaistimuksia" saadaan aikaan mittaamalla virranmuutoksia manipulaattoreissa ja raajoissa. Tämä kuitenkin vaatisi selkeää vastustusta raajojen liikkeille, joten kosketukset ja hipaisut jäisivät huomaamatta. Kuvassa 15 on esimerkki tilanteesta, jossa WorkPartner ei saisi mitään "tuntoaistimuksia", koska laatikko ei vastustaisi puristusta.

Kuva 15. WorkPartner ei saa "tuntoaistimuksia", kun laatikko lipsuu käsistä ja sitä ei voida puristaa.

8.1.2 Oppimiseen liittyviä ongelmia

Tässä työssä on lähdetty toteuttamaan taitoa vaativan tehtävän oppimista. Taitoa vaativan tehtävän oppiminen voidaan luokitella korkean tason oppimiseksi, koska siinä suoritetaan monimutkaisia ja moniosaisia tehtäviä. Yhden mikrotehtävän suorituskaan ei ole taattu onnistumaan, joten kokonaisen tehtäväkokonaisuuden suoritus vaatii WorkPartnerin ohjelmoinnilta, elektroniikalta ja mekaniikalta toimivuutta ja sopeutumiskykyä.

WorkPartnerissa ei ole toteutettu ns. matalan tason oppimista. Matalan tason oppimisessa robottia opetetaan suorittamaan joitakin hyvin yksinkertaisia toimintoja kuten liikkumista oppimalla. Robotti ei aluksi osaa suoriutua toimenpiteestä, kuten kävelemisestä, jolloin se kokeilun ja erehdyksen kautta opettelee kävelemään mahdollisimman tehokkaasti. Tällä tavalla robotti "etsii" omalle rakenteelleen parhaan tavan liikkua. Voisi ajatella, että WorkPartnerin on vaikea suoriutua korkean tason oppimista, jos sille ei ole toteutettu matalankaan tason oppimista. Toisaalta työssä on tullut vastaan osoituksia siitä, että korkean tason oppimisen kautta voitaisiin toteuttaa myös matalamman tason oppimista. WorkPartnerille voidaan lähettää alhaisen tason käskyjä, kuten "liikuta polviniveltä asentoon 10", jolloin voitaisiin kehittää esimerkiksi kävelemiselle sopivat oppimiskriteerit. Tämä havainto vaatii lisätutkimusta.

8.1.3 Ohjelmointiin liittyviä ongelmia

Ohjelmaa (katso liite 1) varten piti tehdä paljon pohjustavaa ohjelmointia, jotta yleensä voitiin aloittaa geneettisen algoritmin, geneettisen koodauksen ja tilakoneen toteutus. Muun muassa huomattava aika meni kuvankäsittely koodin (Jouni Sievilän tunnistusohjelman koodia) sovittamiseen omaan ohjelmaani ja monien alirutiinien, kuten tiedostoon kirjoittamisien, toteuttamiseen.

Ohjelmaa koodatessa ongelmia tuli vastaan geneettisen algoritmin kanssa, kun jokainen mikrotehtävä käyttää erilaisia parametreja kuin muut. Tästä seuraa että geneettiseen algoritmiin piti tehdä monta erilaista alirutiinia, jotta pystyttäisiin käsittelemään (esim. risteyttämään ja tekemään mutaatioita) parametreja. Tavoitteena on muuttaa koodia yleispätevämpään (modulaariseksi) suuntaan, jotta saataisiin yksi GA käsittelemään kaikkia parametreja. Tällöin ohjelmaa laajennettaessa ei tarvitsisi GA:ta laajentaa ollenkaan.

8.2 Tulevaisuuden haasteita

Suurin osa GA:n testauksesta ja suunnittelusta on tapahtunut simuloiduissa tietokoneympäristöissä. (Hagras et al. 2002) Simuloitu ympäristö ei kuitenkaan vastaa täysin todellisuutta, oli se kuinka hyvä tahansa. Todellisessa maailmassa testaus on kyllä hitaampaa kuin simuloidussa ympäristössä, mutta ympäristö voidaan tällöin rakentaa lähes täydellisesti vastaamaan robotin todellista tulevaa työskentely-ympäristöä. Simuloidussa ympäristössä testaus ei ole kuitenkaan turhaa. Simuloinnilla on hyvä tehdä alustavaa kokeilua oppimisen toimivuudesta ja sitten siirtyä testaamaan robottia todellisessa työskentely-ympäristössä. Oppimisen kannalta tärkeimmät testitulokset tulevat juuri todellisessa työskentely-ympäristössä tehdyistä testeistä.

8.3 Jatkotoimenpiteitä

Työssä tehdyssä ohjelmassa pystytään tällä hetkellä opettelemaan yhden taitoa vaativan tehtävän eli "laatikon etsintä ja nosto" –tehtävän suoritusta. Jo nyt tästä ohjelmasta on hyötyä tulevaisuudessa toteutettavien muiden taitoa vaativien tehtävien ohjelmoinnissa, koska sitä voidaan käyttää kätevästi pohjana muiden tekemien tehtävien toteutuksille.

Jotta tästä työstä saataisiin kaikki hyöty irti, olisi suositeltavaa jatkokehittää tätä ohjelmaa, jotta siitä saataisiin tehtyä alusta taitoa vaativien tehtävien oppimiselle. Tällä ohjelmalla voitaisiin kehittää uusia tehtäviä WorkPartnerille, esimerkiksi kuvassa 9 esitetyn graafisen käyttöliittymän tapaan yhdistelemällä graafisia elementtejä. Tämän ohjelman avulla kerättäisiin tietokantaan erilaisia taitoa vaativia tehtäviä, jolloin voitaisiin ennenpitkää tutkia, kuinka monista tehtävistä saataisiin yhdistettyä laajoja ja monimutkaisia tehtävä kokonaisuuksia.

WorkPartnerin ympäristön havainnointia on kehitettävä, mutta miten havainnointi pitäisi toteuttaa, jotta se olisi kätevä, yksinkertainen ja tärkeimmät asiat tulisi otettua huomioon? Jos robotin rakenteeseen liitettäisiin paljon aistinlaitteita lisää ja tallennettaisiin kaikkea mahdollista ympäristöstä saatavaa tietoa, voisi WorkPartner mahdollisesti oppia tulkitsemaan saamaansa palautetta ympäristöstään, tulkitsemalla vain oleellisia asioita. Jos WorkPartner osaisi paremmin analysoida ympäristöään, se pystyisi kenties oppimaan antamaan itse itselleen fitness-kriteerejä. Robotti voisi aluksi seurata ihmisen antamaa opetusta ja tarpeeksi kauan oltuaan ihmisen

opetettavana, se kykenisi tunnistamaan ympäristöstään tehtävän suorituksessa vaadittavat onnistumis-kriteerit ja lopulta oppisi antamaan itselleen palautetta.

Laatikon nostosta voisi tehdä yleispätevämmän, jolloin laatikon ei tarvitse olla tietyssä suunnassa robottiin päin, jotta se voitaisiin nostaa. WorkPartner voisi opetella nostamaan laatikon mistä suunnasta tahansa ja miten päin tahansa. Toisaalta robotti voisi oppia myös suuntaamaan runkonsa ja manipulaattorinsa ajoissa siten, että laatikko voitaisiin nostaa mahdollisimman helposti. Tämä vaatisi WorkPartnerilta laadukasta kuvankäsittelyä, jolloin laatikon värin ja koon lisäksi pitäisi pystyä tunnistamaan laatikon tarkka muoto ja asento robottiin nähden.

WorkPartnerissa voisi olla kannattavaa yrittää toteuttaa matalan tason oppimista, jolloin sitä voitaisiin opettaa liikkumaan (esim. kävelemään) tehokkaasti. Koska WorkPartnerin rakenne on varsin ainutlaatuinen, voitaisiin oppimalla (esim. geneettisen algoritmin avulla) löytää robotin rakenteelle paras tapa liikkua. Matalan tason oppimisella voitaisiin esimerkiksi saada WorkPartner oppimaan, kuinka manipulaattoria siirrettäisiin "eteenpäin". Tällöin päästäisiin pois tarkoista koordinaattipisteiden käytöstä ja lähestyttäisiin enemmän eläimien tapaa liikkua. Esimerkiksi vauva ei aluksi osaa liikuttaa kättään mihinkään järkevään suuntaan. Vauva kokeilun ja erehdyksen kautta oppii liikuttamaan käsiään tarkemmin ja hallitummin, kunnes se oppii saamaan haluamistaan kappaleista otteen.

Oppiminen tuo WorkPartnerin taitoihin suuria etuja, koska ohjelmoijien ei tarvitse kuluttaa aikaa miettiäkseen miten taitoa vaativa tehtävä pitää tarkalleen suorittaa. Riittää, että WorkPartnerille kerrotaan mistä toiminnallisuuksista eli mikrotehtävistä taitoa vaativa tehtävä koostuu, annetaan robotille alkuarvot (satunnaiset tai mietityt) ja annetaan robotin suoritusyrityksille fitness-arvoja. Robotti tekee loput itse.

Oppiminen on ominaisuus, joka on auttanut eläimiä selviytymään ja kehittymään tällä planeetalla. Ihmisetkin ovat kehittäneet kulttuureja, erilaisia kieliä ja monimutkaista tekniikkaa nimenomaan oppimisen ansiosta. Jos roboteille pystytään jo opettamaan yksinkertaisia tehtäviä, on vain ajan kysymys ennen kuin tehtävien taso nousee tarpeeksi korkealle, jolloin roboteista saadaan ihmisille todellinen työkaveri.

Lähdeluettelo

"Animal Learning." Encyclopædia Britannica. 2003. Encyclopædia Britanica Online. Viitattu 18.1.2004. http://search.eb.com/eb/article?eu=109611

Baker, J. E. 1985. Adaptive selection methods for genetic algorithms. In J. J. Grefenstette, ed., *Proceedings of the First International Conference on Genetic Algorithms and Their Applications*. Erlbaum.

De Jong, K. A. 1975. An Analysis of the Behavior of a Class of Genetic Adaptive Systems. Ph.D. thesis, University of Michigan, Ann Arbor.

Eklund, E. 1990. Robotti – ihmisen muotoinen kone. Suomen Tekoälyseuran julkaisu no 5. Ajattelevatko koneet? Tekoäly, tietotekniikka ja robotiikka. Toimittaneet Seppo Stark ja Kati Tyystjärvi. Suomen Tekoälyseura ry. Tiedekeskus Heureka.

Farritor, S., Dubowsky, S. 2001. On Modular Design of Field Robotic Systems. Autonomous Robots 10, 57-65. Kluwer Academic Publishers. Manufactured in the Netherlands.

Fukuda, T., Kubota N. 1997. Adaptation, Learning and Evolution for Intelligent Robotic System. Nagoya University.

Goldberg, D. E. 1990. A note on Boltzmann tournament selection for genetic algorithms and population-oriented simulated annealing. *Complex Systems* 4: 445-460.

Hagras, H., Callaghan, V., Colley, M. 2001. Outdoor Mobile Robot Learning and Adaptation. IEEE Robotics & Automation Magazine. September 2001, 53-69.

Hagras, H., Colley, M., Callaghan, V., Carr-West, M. 2002. Online Learning and Adaptation of Mobile Robots for Sustainable Agriculture. Autonomous Robots 13, 37-52. Kluwer Academic Publishers. Manufactured in the Netherlands.

Halme, A., Sievilä, J., Kauppi, I., Ylönen, S. 2003. Performing skilled work with an interactively operated service robot. The 4th International Conference on Field and Service Robotics.

Halme, J. 1997. Utilization of genetic algorithm in online tuning of fluid power servos. Lappeenranta University of Technology. Research Papers No. 60.

Husbands, P., Harvey, I., Cliff, D., Miller, G. 1997. Artificial Evolution: A New Path for Artificial Intelligence? Brain and Cognition 34, 130-159.

Kubota, N., Morioka, T., Kojima, F., Fukuda, T. 2001. Learning of mobile robots using perception-based genetic algorithm. Measurement 29, 237-248. Elsevier.

Leppänen, I., Salmi, S., Halme, A. 1998. WorkPartner – HUT-Automation's New Hybrid Walking Machine. CLAWAR'98. Brussels. 1st International Conference of Climbing and Walking Robots.

Mitchell, Melanie. 1996. An Introduction to Genetic Algorithms. A Bradford Book. The MIT Press. Cambridge, Massachusetts.

Nolfi, S., Floreano, D. 1999. Learning and Evolution. Autonomous Robots 7, 89-113. Kluwer Academic Publishers. Manufactured in the Netherlands.

Ogino, M., Katoh, Y., Aona, M., Asada, M., Hosoda, K. 2003.Vision-based reinforcement learning for humanoid behavior generation with rhythmic walking parameters. Proceedings of 2003 IEEE/RSJ International Conference on Intelligent Robots and Systems. Pp. 1665-1671.

Oksanen, T. 2003. Nelijalkainen nelipyöräinen robotti liikkuvana systeeminä. Helsinki University of Technology, Automation Laboratory. Series A: Research Reports No. 25. 2003.

Robbins, S. P. 2001. *Organizational behavior*, Prentice-Hall. New Jersey.

Russell, S., Norvig, P. 1995. Artificial Intelligence, A Modern Approach. Second edition. Prentice Hall. By Pearson Education, Inc.

Schaffer, J. D., Caruana, R. A., Eshelman, L. J. and Das, R. 1989. A study of control parameters affecting online performance of genetic algorithms for function optimization. In J. D. Schaffer, ed., *Proceedings of the Third International Conference on Genetic Algorithms*. Morgan Kaufmann.

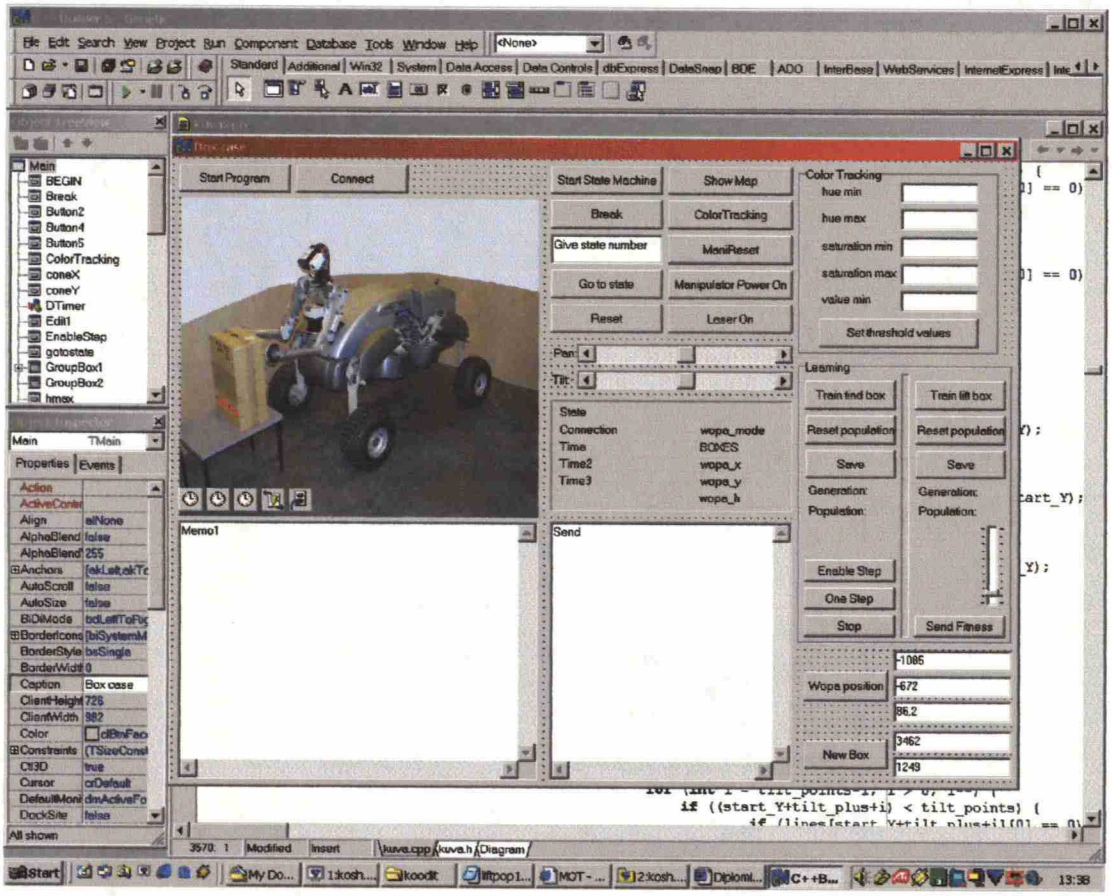
Seppänen, J. 1990. Mitä on ajattelu? Suomen Tekoälyseuran julkaisu no 5. Ajattelevatko koneet? Tekoäly, tietotekniikka ja robotiikka. Toimittaneet Seppo Stark ja Kati Tyystjärvi. Suomen Tekoälyseura ry. Tiedekeskus Heureka.

Sievilä, J. 2003. Palvelurobotin suorittama etsintä- ja tunnistustehtävä. Teknillinen Korkeakoulu, Automaatio- ja systeemitekniikan osasto. Diplomityö. 60 p.

Sinkkonen, I., Kuoppala, H., Parkkinen, J., Vastamäki, R. 2002. Käytettävyyden psykologia. Edita Publishing Oy. IT Press.

Liite 1

Diplomityössä tehty ohjelma toteutettiin C++ Builder 6 ohjelmointialustalla. Ohjelman käyttöliittymässä on kaksi selkeästi erillistä osaa. Nimittäin kuvankäsittelyosuus, joka pohjautuu Jouni Sievilän tekemään tunnistusohjelmaan (Sievilä 2003) ja oppimisosuus. Oppimisosuudessa hallitaan kahden eri osatehtävän ("laatikon etsintä" ja "laatikon nosto") oppimista. Oppimiseen liittyvät ohjaukset näkyvät alla olevassa kuvassa 16 oikeassa reunassa rajattuna "Learning" nimisellä kehyksellä.



Kuva 16. Työpöydän näkymä Borland C++ Builder ohjelmasta.